Sun xVM VirtualBox®

Version 1.6.2

http://www.virtualbox.org

May 31, 2008

Contents

1 Introduction

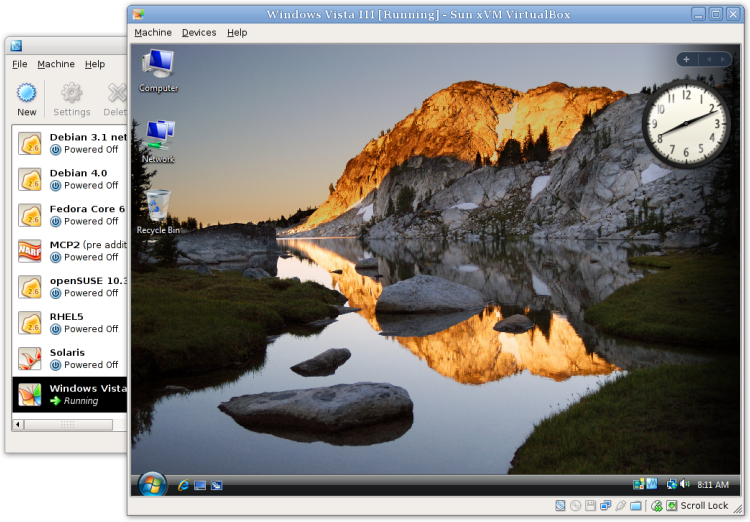
Sun xVM VirtualBox is a family of virtual machine products targeting desktop computers, enterprise servers and embedded systems. With VirtualBox, 32-bit operating systems for Intel and AMD processors can be virtualized on such 32-bit as well as 64-bit machines (see chapter 1.3.1, *Supported host operating systems*, page 12).

You can find a brief feature overview in chapter 1.2, *Features overview*, page 10; see chapter 12, *Change log*, page 149 for a detailed list of version changes.

1.1 Virtualization basics

With VirtualBox, you can run unmodified operating systems – including all of the software that is installed on them – directly on top of your existing operating system, in a special environment that is called a "virtual machine". Your physical computer is then usually called the "host", while the virtual machine is often called a "guest".

The following image shows you how VirtualBox, on a Linux host, is running Windows Vista as guest operating system in a virtual machine window:



VirtualBox allows the guest code to run unmodified, directly on the host computer, and the guest operating system "thinks" it's running on real machine. In the background, however, VirtualBox intercepts certain operations that the guest performs to make sure that the guest does not interfere with other programs on the host.

The techniques and features that VirtualBox provides are useful for several scenarios:

- **Operating system support.** With VirtualBox, one can run software written for one operating system on another (for example, Windows software on Linux) without having to reboot to use it. You can even install in a virtual machine an old operating system such as DOS or OS/2 if your real computer's hardware is no longer supported.

- **Infrastructure consolidation.** Virtualization can significantly reduce hardware and electricity costs. The full performance provided by today's powerful hardware is only rarely really needed, and typical servers have an average load of only a fraction of their theoretical power. 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. With VirtualBox, you can even run virtual machines as mere servers for the VirtualBox Remote Desktop Protocol (VRDP), with full client USB support. This allows for consolidating the desktop machines in an enterprise on just a few RDP servers, while the actual clients will only have to be able to display VRDP data any more.

- **Testing and disaster recovery.** Once installed, a virtual box and its virtual hard disk 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 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 (e.g. after installing misbehaving software or infecting the guest with a virus), one can easily switch back to a previous snapshot and avoid the need of frequent backups and restores.

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): the operating system of the physical computer where VirtualBox is running. There are versions of VirtualBox for several host operating systems (see chapter 1.3.1, *Supported host operating systems*, page 12 for further information). While the various VirtualBox versions are usually discussed together in this document, there may be platform-specific differences which we will point out where appropriate.

Guest operating system (guest OS): the operating system that is running inside the virtual machine. Theoretically, VirtualBox can run any x86 operating system (DOS, Windows, OS/2, FreeBSD, 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 operating systems. So while your

favorite operating system *may* run as a guest, we officially support and optimize for a select few (which, however, include the most common ones).

See chapter 1.3.2, *Supported guest operating systems*, page 13 for further information.

Virtual machine (VM). When running, a VM is the special environment that VirtualBox creates for your guest operating system. So, in other words, you run your guest operating system "in" a VM. Normally, a VM will be shown as a window on your computer's desktop, but depending on which of the various frontends of VirtualBox you use, it can be displayed in full-screen mode or remotely by use of the Remote Desktop Protocol (RDP).

Sometimes we also use the term "virtual machine" in a more abstract way. Internally, VirtualBox thinks of a VM as a set of parameters that determine its operation. These settings are mirrored in the VirtualBox graphical user interface as well as the `VBoxManage` command line program; see chapter 8, *VBoxManage reference*, page 96. They include hardware settings (how much memory the VM should have, what hard disks VirtualBox should virtualize through which container files, what CD-ROMs are mounted etc.) as well as state information (whether the VM is currently running, saved, its snapshots etc.).

In other words, a VM is also what you can see in its settings dialog.

Guest Additions. With "Guest Additions", we refer to special software packages that are shipped with VirtualBox. Even though they are part of VirtualBox, they are designed to be installed *inside* a VM to improve performance of the guest OS and to add extra features. This is described in detail in chapter 4, *The VirtualBox Guest Additions*, page 52.

1.2 Features overview

Here's a brief outline of VirtualBox's main features:

- **Clean architecture; unprecedented modularity.** 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 VirtualBox graphical user interface and then control that machine from the command line, or even remotely. See chapter 7, *Alternative front-ends; remote virtual machines*, page 87 for details.

  Due to its modular architecture, VirtualBox can also expose its full functionality and configurability through a comprehensive **software development kit (SDK),** which allows for integrating every aspect of VirtualBox with other software systems. Please see chapter 10, *VirtualBox programming interfaces*, page 127 for details.

- **No hardware virtualization required.** VirtualBox does not require processor features built into newer hardware like VT-x (on Intel processors) or AMD-V (on AMD processors). As opposed to many other virtualization solutions, you can therefore use VirtualBox even on older hardware where these features are not present. In fact, VirtualBox's sophisticated software techniques are typically faster than hardware virtualization, although it is still possible to enable hardware virtualization on a per-VM basis. Only for some exotic guest operating systems like OS/2, hardware virtualization is required.

- **Easy portability.** VirtualBox runs on a large number of 32-bit and 64-bit host operating systems (Windows, Linux, Mac OS X and Solaris), and new platforms are added regularly (see chapter 1.3.1, *Supported host operating systems*, page 12).

- **Guest Additions.** The 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 and more. The Guest Additions are described in detail in chapter 4, *The VirtualBox Guest Additions*, page 52.

  In particular, Guest Additions provide for "shared folders", which let you access files from the host system from within a guest machine. Shared folders are described in chapter 4.5, *Folder sharing*, page 59.

- **XML configuration store.** VirtualBox stores all its configuration in XML files: one XML document for global settings and a XML file per virtual machine. This allows you to transport VM definitions between the different frontends and even across host computers. For details, please refer to chapter 9.1, *VirtualBox configuration data*, page 114.

- **Great hardware support.** Among others, VirtualBox supports:

  - **Full ACPI support.** The Advanced Configuration and Power Interface (ACPI) is fully supported by VirtualBox. This eases cloning of PC images from real machines or third-party virtual machines into VirtualBox. With its unique **ACPI power status support,** VirtualBox can even report to ACPI-aware guest operating systems 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 (e.g. in fullscreen modes).
  - **I/O APIC support.** VirtualBox virtualizes an Input/Output Advanced Programmable Interrupt Controller (I/O APIC) which is found in many modern PC systems. This eases cloning of PC images from real machines or 3rd party virtual machines into VirtualBox.
  - **USB device support.** VirtualBox implements a virtual USB controller and allows 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. For details, see chapter 3.7.6.1, *USB settings*, page 48.

- **Multiscreen resolutions.** 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 allows 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. For details, see chapter 5.5, *iSCSI servers*, page 66.
- **PXE Network boot.** The integrated virtual network cards of VirtualBox fully support remote booting via the Preboot Execution Environment (PXE).

- **Multigeneration snapshots.** VirtualBox can save successive snapshots of the state of the virtual machine. You can revert the virtual machine to the state of any of the snapshots. For details, see chapter 3.4.4, *Snapshots*, page 38.

- **VRDP remote access.** You can run any virtual machine in a special VirtualBox program that acts as a server for the VirtualBox Remote Desktop Protocol (VRDP). With this unique feature, VirtualBox provides high-performance remote access to any virtual machine. A custom RDP server has been built directly into the virtualization layer and offers unprecedented performance and feature richness.

  VRDP support is described in detail in chapter 7.4, *Remote virtual machines (VRDP support)*, page 90.

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

  - **Extensible RDP authentication.** VirtualBox already supports Winlogon on Windows and PAM on Linux for RDP authentication. In addition, it includes an easy-to-use SDK which allows you to create arbitrary interfaces for other methods of authentication; see chapter 9.3, *Custom external VRDP authentication*, page 116 for details.
  - **USB over RDP.** Via RDP virtual channel support, VirtualBox also allows you to connect arbitrary USB devices locally to a virtual machine which is running remotely on a VirtualBox RDP server; see chapter 7.4.3, *Remote USB*, page 93 for details.

1.3 Operating system support

1.3.1 Supported host operating systems

Currently, VirtualBox is available for the following host operating systems:

- **Windows** hosts:
  - Windows XP, all service packs (32-bit)
  - Windows Server 2003 (32-bit)
  - Windows Vista (32-bit and 64-bit[1]).
- **Apple Mac OS X** hosts (Intel hardware only, all versions of Mac OS X supported)[2]
- **Linux** hosts (32-bit and 64-bit[3]):
  - Debian GNU/Linux 3.1 ("sarge") and 4.0 ("etch")
  - Fedora Core 4 to 8
  - Gentoo Linux
  - Redhat Enterprise Linux 3, 4 and 5
  - SUSE Linux 9 and 10, openSUSE 10.1, 10.2 and 10.3
  - Ubuntu 5.10 ("Breezy Badger"), 6.06 ("Dapper Drake"), 6.10 ("Edgy Eft"), 7.04 ("Feisty Fawn"), 7.10 ("Gutsy Gibbon")
  - Mandriva 2007.1 and 2008.0

  It should be possible to use VirtualBox on most systems based on Linux kernel 2.4 or 2.6 using either the VirtualBox installer or by doing a manual installation; see chapter 2.3, *Installing on Linux hosts*, page 17.
- **Solaris** hosts (32-bit and 64-bit[4]):
  - OpenSolaris ("Nevada" build 81 and higher)
  - Solaris 10 (u4 and higher, experimental support)

1.3.2 Supported guest operating systems

While VirtualBox is designed to provide a generic virtualization environment for x86 systems, our focus is to optimize the product's performance for a select list of guest systems. The following table provides an overview of current support:

Windows NT 4.0 All versions/editions and service packs are fully supported; however, there are some issues with older service packs. We recommend to install service pack 6a. Guest Additions are available with a limited feature set.

Windows 2000 / XP / Server 2003 / Vista All versions/editions and service packs are fully supported. Guest Additions are available.

[1] Support for 64-bit Windows was added with VirtualBox 1.5.
[2] Preliminary Mac support (beta stage) had been added with VirtualBox 1.4, full support with 1.6.
[3] Support for 64-bit Linux was added with VirtualBox 1.4.
[4] Support for OpenSolaris was added with VirtualBox 1.6.

DOS / Windows 3.x / 95 / 98 / ME Limited testing has been performed. Use beyond legacy installation mechanisms not recommended. No Guest Additions available.

Linux 2.4 Limited support.

Linux 2.6 All versions/editions are fully supported. We strongly recommend using version 2.6.13 or higher for better performance. However, version 2.6.18 (and some 2.6.17 versions) introduced a race condition that can cause boot crashes in VirtualBox; if you must use a kernel >= 2.6.17, we advise to use 2.6.19 or later. Guest Additions are available.

Solaris 10, OpenSolaris Fully supported, Guest Additions are available with a limited feature set.[5]

FreeBSD Limited support. Guest Additions are not available yet.

OpenBSD Versions 3.7 and 3.8 are supported. Guest Additions are not available yet.

OS/2 Warp 4.5 Requires hardware virtualization to be enabled. We officially support MCP2 only; other OS/2 versions may or may not work. Guest Additions are available with a limited feature set.[6]

[5] See chapter 13, *Known issues*, page 180.
[6] See chapter 13, *Known issues*, page 180.

2 Installation

As installation of VirtualBox varies depending on your host operating system, we provide installation instructions in three separate chapters for Windows, Linux and OpenSolaris, respectively.

2.1 Installing on Windows hosts

2.1.1 Prerequisites

For the various versions of Windows that we support as host operating systems, please refer to chapter 1.3.1, *Supported host operating systems*, page 12.

In addition, Windows Installer 1.1 or higher must be present on your system. This should be the case if you have all recent Windows updates installed.

2.1.2 Performing the installation

The VirtualBox installation can be started

- either by double-clicking on its Microsoft Installer archive (MSI file)
- or by entering

  ```
  msiexec /i VirtualBox.msi
  ```

  on the command line.

In either case, this will display the installation welcome dialog and allow you to choose where to install VirtualBox to and which components to install. In addition to the VirtualBox application, the following components are available:

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

Networking This package contains extra networking drivers for your Windows host that VirtualBox needs to support Host Interface Networking (to make your VM's virtual network cards accessible from other machines on your physical network).

Depending on your Windows configuration, you may see warnings about "unsigned drivers" or similar. Please select "Continue" on these warnings as otherwise VirtualBox might not function correctly after installation.

The installer will create a "VirtualBox" group in the programs startup folder which allows you to launch the application and access its documentation.

With standard settings, VirtualBox will be installed for all users on the local system. In case this is not wanted, you have to invoke the installer as follows:

```
msiexec /i VirtualBox.msi ALLUSERS=2
```

This will install VirtualBox only for the current user.

2.1.3 Uninstallation

As we use the Microsoft Installer, VirtualBox can be safely uninstalled at any time by choosing the program entry in the "Add/Remove Programs" applet in the Windows Control Panel.

2.1.4 Unattended installation

Unattended installations can be performed using the standard MSI support.

2.2 Installing on Mac OS X hosts

2.2.1 Performing the installation

For Mac OS X hosts, VirtualBox ships in a disk image (dmg) file. Perform the following steps:

1. Double-click on that file to have its contents mounted.

2. A window will open telling you to double click on the `VirtualBox.mpkg` installer file displayed in that window.

3. This will start the installer, which will allow you to select where to install VirtualBox to.

After installation, you can find a VirtualBox icon in the "Applications" folder in the Finder.

2.2.2 Uninstallation

To uninstall VirtualBox, open the disk image (dmg) file again and double-click on the uninstall icon contained therein.

2.2.3 Unattended installation

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

Mount the disk image (dmg) file as described in the normal installation. Then open a terminal session and execute:

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

2.3 Installing on Linux hosts

2.3.1 Prerequisites

For the various versions of Linux that we support as host operating systems, please refer to chapter 1.3.1, *Supported host operating systems*, page 12.

You will 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 VirtualBox):

- Qt 3.3.5 or higher;
- SDL 1.2.7 or higher (this graphics library is typically called `libsdl` or similar).

Note: To be precise, these packages are only required if you want to run the VirtualBox graphical user interfaces. In particular, `VirtualBox`, our main graphical user interface, requires both Qt and SDL; `VBoxSDL`, our simplified GUI, requires only SDL. By contrast, if you only want to run the headless VRDP server that comes with VirtualBox, neither Qt nor SDL are required.

2.3.2 The VirtualBox kernel module

Most people who install VirtualBox using packages specially created for their Linux distribution will be able to safely skip this section unless they run into problems during the installation. All that they will need to remember is that they may have to reinstall VirtualBox (or recreate the VirtualBox kernel module by running

```
/etc/init.d/vboxdrv setup
```

as root) if the Linux kernel on their system is updated.

VirtualBox uses a special kernel module to perform physical memory allocation and to gain control of the processor for guest system execution. Without this kernel module, you will still be able to work with Virtual Machines in the configuration interface, but you will not be able to start any virtual machines.

To be able to install this kernel module, you will have to prepare your system for building external kernel modules. As this process can vary from system to system, we will only describe what to do for systems we have tested

- Most Linux distributions can be set up simply by installing the right packages. Normally, these will be the GNU compiler (GCC), GNU Make (make) and packages containing header files for your kernel. *The version numbers of the header file packages must be the same as that of the kernel you are using.*

  - For recent Linux distributions (for example Fedora Core 5 and later, Ubuntu 7.10 (Gutsy) and later and Mandriva 2007.1 and later) we recommend installing DKMS. This framework helps to build kernel modules and to deal with kernel upgrades.
  - In newer Debian and Ubuntu releases, you must install the right version of the `linux-headers` and if it exists the `linux-kbuild` package. Current Ubuntu releases should have the right packages installed by default.
  - In older Debian and Ubuntu releases, you must install the right version of the `kernel-headers` package.
  - On Fedora and Redhat systems, the package is `kernel-devel`.
  - On SUSE and OpenSUSE Linux, you must install the right versions of the `kernel-source` and `kernel-syms` packages.

- Alternatively, if you built your own kernel `/usr/src/linux` will point to your kernel sources, and you have not removed the files created during the build process, then your system will already be correctly set up.

2.3.3 USB and advanced networking support

In order to use VirtualBox's USB support, the user account under which you intend to run VirtualBox must have read and write access to the USB filesystem (`usbfs`).

In addition, access to `/dev/net/tun` will be required if you want to use Host Interface Networking, which is described in detail in chapter 6.5, *Introduction to Host Interface Networking (HIF)*, page 73.

2.3.4 Performing the installation

VirtualBox is available in a number of package formats native to various common Linux distributions (see chapter 1.3.1, *Supported host operating systems*, page 12 for details). In addition, there is an alternative generic installer (.run) which should work on most Linux distributions.

2.3.4.1 Installing VirtualBox from a Debian/Ubuntu package

First, download the appropriate package for your distribution. The following examples assume that you are installing to an Ubuntu Edgy system. Use `dpkg` to install the Debian package:

```
sudo dpkg -i VirtualBox_1.6.2_Ubuntu_edgy.deb
```

You will be asked to accept the VirtualBox Personal Use and Evaluation License. Unless you answer "yes" here, the installation will be aborted.

The group `vboxusers` will be created during installation. Note that a user who is going to run VirtualBox must be member of that group. A user can be made member of the group `vboxusers` through the GUI user/group management or at the command line with

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

Also note that adding an active user to that group will require that user to log out and back in again. This should be done manually after successful installation of the package.

The installer will also search for a VirtualBox kernel module suitable for your kernel. The package includes pre-compiled modules for the most common kernel configurations. If no suitable kernel module is found, the installation script tries to build a module itself. If the build process is not successful you will be shown a warning and the package will be left unconfigured. Please have a 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 chapter 2.3.2, *The VirtualBox kernel module*, page 17). After correcting any problems, do

```
sudo /etc/init.d/vboxdrv 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 chapter 11.5.1, *Linux kernel module refuses to load*, page 145 for further information.

Once VirtualBox has been successfully installed and configured, you can start it by selecting "VirtualBox" in your start menu or from the command line (see chapter 2.3.5, *Starting VirtualBox on Linux*, page 23).

2.3.4.2 Using the alternative installer (VirtualBox.run)

The alternative installer performs the following steps:

- It unpacks the application files to a target directory of choice. By default, the following directory will be used:

  ```
  /opt/VirtualBox-1.6.2
  ```

- It builds the VirtualBox kernel module (`vboxdrv`) and installs it.
- It creates `/etc/init.d/vboxdrv`, an init script to start the VirtualBox kernel module.
- It creates a new system group called `vboxusers`.
- It creates symbolic links to `VirtualBox`, `VBoxSDL`, `VBoxVRDP`, `VBoxHeadless` and `VBoxManage` in `/usr/bin`.
- It creates `/etc/udev/60-vboxdrv.rules`, a description file for udev, if that is present, which makes the module accessible to anyone in the group `vboxusers`.
- It 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. If you do not want the installer to ask you whether you wish to accept the license agreement (for example, for performing unattended installations), you can add the parameter `license_accepted_unconditionally`. Finally, if you want to use a directory other than the default installation directory, add the desired path as an extra parameter.

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

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

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

After that you need to put every user which should be able to use VirtualBox in the group `vboxusers`, either through the GUI user management tools or by running 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, e.g. like this: `usermod -G group1,group2,vboxusers username`.

If any users on your system should be able to access host USB devices from within VirtualBox guests, you should also add them to the appropriate user group that your distribution uses for USB access, e.g. `usb` or `usbusers`.

2.3.4.3 Performing a manual installation

If, for any reason, you cannot use the shell script installer described previously, you can also perform a manual installation. Invoke the installer like this:

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

This will unpack all the files needed for installation in the directory `install` under the current directory. The 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
```

or as root:

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

The sources for VirtualBox's kernel module are provided in the `src` directory. To build the module, change to the directory and issue

```
make
```

If everything builds correctly, issue 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 perform

```
make install
```

The 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 user who is going to run VirtualBox needs read and write permission on the VirtualBox kernel module device node `/dev/vboxdrv`. You can either define a `vboxusers` group by entering

```
groupadd vboxusers
chgrp vboxusers /dev/vboxdrv
chmod 660 /dev/vboxdrv
```

or, alternatively, simply give all users access (insecure, not recommended!)

```
chmod 666 /dev/vboxdrv
```

You should also add any users who will be allowed to use host USB devices in VirtualBox guests to the appropriate USB users group for your distribution. This group is often called `usb` or `usbusers`.

Next, you will have to install the system initialization script for the kernel module:

```
cp /opt/VirtualBox/vboxdrv.sh /etc/init.d/vboxdrv
```

(assuming you installed VirtualBox to the `/opt/VirtualBox` directory) and activate the initialization script using the right method for your distribution. You should create VirtualBox's configuration file:

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

and, for convenience, create the following symbolic links:

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

2.3.4.4 Updating and uninstalling VirtualBox

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

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

or as root

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

To manually uninstall VirtualBox, simply undo the steps in the manual installation in reverse order.

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

These default settings can be applied with

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

prior to the installation of the VirtualBox Debian package.

2.3.4.6 Automatic installation of .rpm packages

The .rpm format does not provide a configuration system comparable to the debconf system. To configure the installation process of our .rpm packages, a file `/etc/default/virtualbox` is interpreted. The automatic generation of the udev rule can be prevented by the following setting:

```
INSTALL_NO_UDEV=1
```

The creation of the group vboxusers can be prevented by

```
INSTALL_NO_GROUP=1
```

If the line

```
INSTALL_NO_VBOXDRV=1
```

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

2.3.5 Starting VirtualBox on Linux

The easiest way to start a VirtualBox program is by running the program of your choice (`VirtualBox`, `VBoxManage`, `VBoxSDL` 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 VirtualBox without installing it first. You should start by compiling the `vboxdrv` kernel module (see above) and inserting it into the Linux kernel. VirtualBox consists of a service daemon (`VBoxSVC`) and several application programs. The daemon is automatically started if necessary. All 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 VirtualBox applications (`VirtualBox`, `VBoxSDL`, `VBoxManage` and `VBoxHeadless`) require the VirtualBox directory to be in the library path:

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

2.4 Installing on OpenSolaris hosts

For the various versions of OpenSolaris that we support as host operating systems, please refer to chapter 1.3.1, *Supported host operating systems*, page 12.

If you have a previously installed instance of VirtualBox on your OpenSolaris host, please uninstall it first before installing a new instance. Refer to chapter 2.4.3, *Uninstallation*, page 25 for uninstall instructions.

2.4.1 Performing the installation

VirtualBox is available as a standard Solaris package. Download the appropriate package for your system. **The installation must be performed as root and from the global zone** as the 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. Execute the following commands:

```
gunzip -cd VirtualBox-1.6.2-SunOS-x86.tar.gz | tar xvf -
```

Starting with VirtualBox 1.6.2 we ship the VirtualBox kernel interface module (vbi). The purpose of this module is to shield the VirtualBox kernel driver from changes to the SunOS kernel. If you do not have vbi already installed (check for the existence of the file `/platform/i86pc/kernel/misc/vbi`) install it by executing the command:

```
pkgadd -G -d VirtualBoxKern-1.6.2-SunOS.pkg
```

Future versions of OpenSolaris may ship the VirtualBox kernel interface module, in which case you can remove this one before upgrading OpenSolaris.

Next you should install the main VirtualBox package using:

```
pkgadd -d VirtualBox-1.6.2-SunOS-x86.pkg
```

> **Note:** If you are using Solaris Zones, to install VirtualBox only into the current zone and not into any other zone, use `pkgadd -G`. For more information refer to the `pkgadd` manual; see also chapter 2.4.5, *Configuring a zone for running VirtualBox*, page 26.

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 VirtualBox kernel module. Following this confirmation the installer will install 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. VirtualBox would be installed in `/opt/VirtualBox`.

2.4.2 Starting VirtualBox on OpenSolaris

The easiest way to start a VirtualBox program is by running the program of your choice (`VirtualBox`, `VBoxManage`, `VBoxSDL` 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 type the full path.

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

Note: Solaris 10 does not ship the `libdlpi.so` library required by VirtualBox for Host Interface Networking (refer chapter 6.5, *Introduction to Host Interface Networking (HIF)*, page 73), hence Host Interface Networking is currently not supported for Solaris 10 hosts.

2.4.3 Uninstallation

Uninstallation of VirtualBox on OpenSolaris requires root permissions. To perform the uninstallation, start a root terminal session and execute:

```
pkgrm SUNWvbox
```

After confirmation, this will remove VirtualBox from your system.

To uninstall the VirtualBox kernel interface module, execute:

```
pkgrm SUNWvboxkern
```

2.4.4 Unattended installation

To perform a non-interactive installation of VirtualBox we have provided a response file named `autoresponse` that the installer will use for responses to inputs rather than ask them from you.

Extract the tar.gz package as described in the normal installation. Then open a root terminal session and execute:

```
pkgadd -d VirtualBox-1.6.2-SunOS-x86 -n -a autoresponse SUNWvbox
```

To perform a non-interactive uninstallation, open a root terminal session and execute:

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

2.4.5 Configuring a zone for running VirtualBox

Starting with VirtualBox 1.6 it is possible to run VirtualBox from within Solaris zones. For an introduction of Solaris zones, please refer to http://www.sun.com/bigadmin/features/articles/solaris_zones.jsp.

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

```
zonecfg -z vboxzone
```

Inside the `zonecfg` prompt add the `device` resource and `match` properties to the zone. Here's how it can be done:

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

Replace "vboxzone" with the name of the zone in which you intend to run VirtualBox. Next reboot the zone using `zoneadm` and you should be able to run VirtualBox from within the configured zone.

3 Starting out with VirtualBox

As already mentioned in chapter 1.1, *Virtualization basics*, page 8, VirtualBox allows you to run each of your guest operating systems on its own virtual computer system. The guest system will run in its virtual machine (VM) as if it were installed on a real computer, according to the settings of the virtual system you have created for it. All software running on the guest system does so as it would on a real machine.

You have considerable latitude in deciding what virtual hardware will be provided to the guest. The virtual hardware can be used for communicating with the host system or with other guests. For instance, if you provide VirtualBox with the image of a CD-ROM in an ISO file, VirtualBox can present this image to a guest system as if it were a physical CD-ROM. Similarly, you can give a guest system access to the real network via its virtual network card, and, if you choose, give the host system, other guests, or computers on the internet access to the guest system.

VirtualBox comes with many advanced interfaces, which will be described later in this manual:

- chapter 7.3, *VBoxSDL, the simplified VM displayer*, page 89 explains how to run a single VM at a time with a reduced graphical interface;
- chapter 7.4.1, *VBoxHeadless, the VRDP-only server*, page 91 shows how to run virtual machines remotely;
- chapter 8, *VBoxManage reference*, page 96 explains how to use create, configure, and control virtual machines completely from the command line.

The following introductory sections, however, describe `VirtualBox`, the graphical user interface, which is the simplest way to get started.

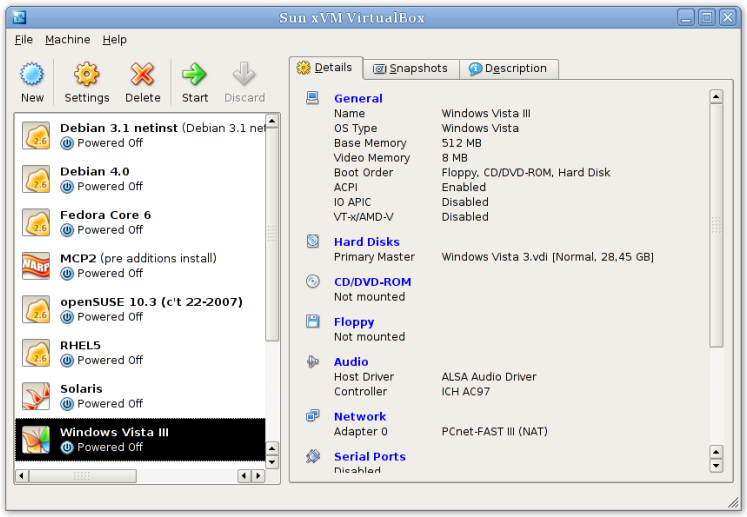
3.1 Starting the graphical user interface

After installing VirtualBox as described in chapter 2, *Installation*, page 15, on Windows, you can start VirtualBox as follows:

- On a Windows host, in the standard “Programs” menu, click on the item in the “VirtualBox” group.
- 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 Solaris host, depending on your desktop environment, a "VirtualBox" item may have been placed in either the "System" or "System Tools" group of your "Applications" menu. Alternatively, you can type `VirtualBox` in a terminal.

A window like the following should come up:

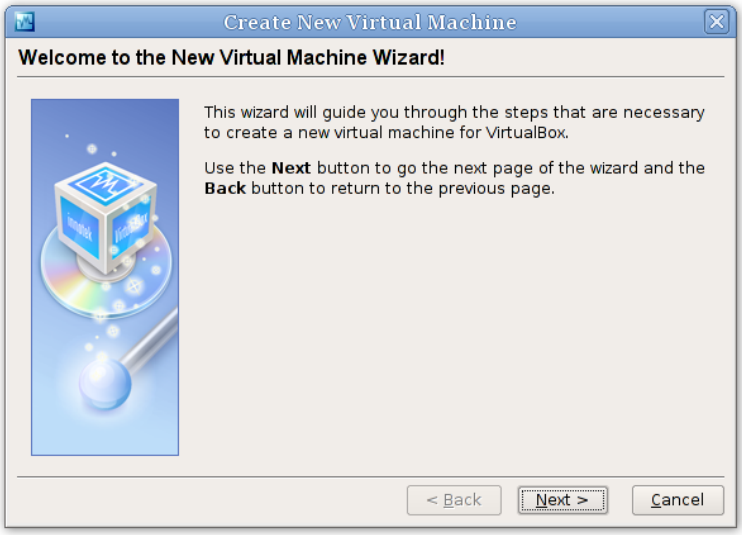


On the left, you can see a pane that lists all the virtual machines you have created so far (three in the example above). A row of buttons above it allows you to create new VMs and work on existing VMs. The pane on the right displays the properties of the virtual machine currently selected, if any.

When you start VirtualBox for the first time, as there is no virtual machine yet, everything will be empty.

3.2 Creating a virtual machine

Clicking on the "New" button in the user interface will guide you through setting up a new virtual machine (VM). A wizard will show up:



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. A **name** for your VM, and the **type of operating system** (OS) you want to install.

   The name is what you will later see in the VirtualBox main window, and what your settings will be stored under. It is purely informational, but once you have created a few VMs, you will appreciate if you have given your VMs informative names. "My VM" probably is therefore not as useful as "Windows XP SP2".

   For "Operating System Type", select the operating system that you want to install later. While this setting presently has no lasting effect, VirtualBox will use this setting to display an operating system accordingly and also make certain recommendations later based on your selection (such as the amount of memory and hard disk space to allocate), and future VirtualBox versions may offer certain system-specific virtualization features. It is therefore recommended to always set it to the correct value.

2. The **amount of memory (RAM)** that the virtual machine should have for itself. Every time a virtual machine is started, VirtualBox will allocate this much memory from your host machine and present it to the guest operating system, which will report this size as the (virtual) computer's installed RAM.

Note: 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.

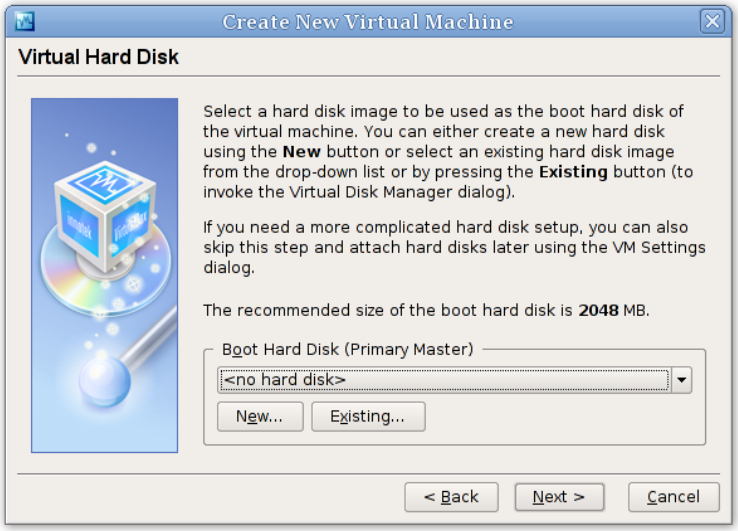
A Windows XP guest will require at least a few hundred MB RAM to run properly, and Windows Vista will even refuse to install with less than 512 MB. Of course, if you want to run graphics-intensive applications in your VM, you may require even more RAM.

So, as a rule of thumb, if you have 1 GB of RAM or more in your host computer, it is usually safe to allocate 512 MB to each VM. But, in any case, make sure you always have at least 256-512 MB of RAM left on your host operating system. Otherwise you may cause your host OS to excessively swap out memory to your hard disk, effectively bringing your host system to a standstill.

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

3. Next, you must specify a **virtual hard disk** for your VM. There are several ways in which VirtualBox can provide hard disk space to a VM, but the most common way is to use a large image file on your "real" hard disk, whose contents VirtualBox presents to your VM as if it were a complete hard disk.

   The wizard presents to you the following window:



The wizard allows you to create an image file or use an existing one. Note also that the disk images can be separated from a particular VM, so even if you delete a VM, you can keep the image, or copy it to another host and create a new VM for it there.

In the wizard, you have the following options:

- If you have previously created any virtual hard disks which have not been attached to other virtual machines, you can select those from the drop-down list in the wizard window.
- Otherwise, to create a new virtual hard disk, press the **"New"** button.
- Finally, for more complicated operations with virtual disks, the **"Existing..."** button will bring up the Virtual Disk Manager, which is described in more detail in chapter 3.5, *The Virtual Disk Manager*, page 39.

Most probably, if you are using VirtualBox for the first time, you will want to create a new disk image. Hence, press the "New" button.

This brings up another window, the **"Create New Virtual Disk Wizard".**

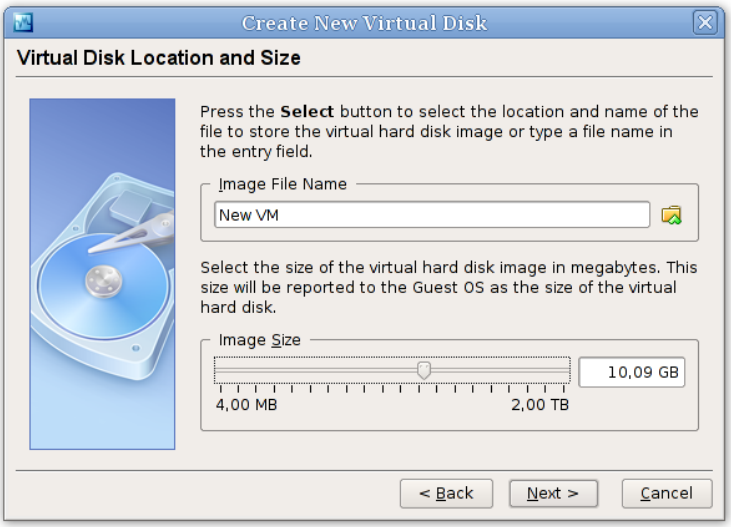
VirtualBox supports two types of image files:

- A **dynamically expanding file** will only grow in size when the guest actually stores data on its virtual hard disk. It will therefore initially be small on the host hard drive and only later grow to the size specified as it is filled with data.
- A **fixed-size file** will immediately occupy the file specified, even if only a fraction of the 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 expanding file.

For details about the differences, please refer to chapter 5.2, *Virtual Disk Image (VDI) files*, page 64.

To prevent your physical hard disk from running full, VirtualBox limits the size of the image file. Still, it needs to be large enough to hold the contents of your operating system and the applications you want to install – for a modern Windows or Linux guest, you will probably need several gigabytes for any serious use:



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

4. After clicking on **"Finish"**, your new virtual machine will be created. You will then see it in the list on the left side of the main window, with the name you have entered.

3.3 Basics of virtual machine configuration

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

Clicking on the "Settings" button in the toolbar at the top of VirtualBox main window brings up a detailed window where you can configure many of the properties of the VM that is currently selected. But be careful: even though it is possible to change all

VM settings after installing a guest operating system, certain changes might prevent a guest operating system 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 simply because the settings dialog allows you to change fundamental characteristics of the virtual computer that is created for your guest operating system, and this operating system may not take it well when, for example, half of its memory is taken away from under its feet. As a result, if the "Settings" button is disabled, shut down the current VM first.

VirtualBox provides a plethora 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 3.7, *Virtual machine settings*, page 41. Even more parameters are available with the command line interface; see chapter 8, *VBoxManage reference*, page 96.

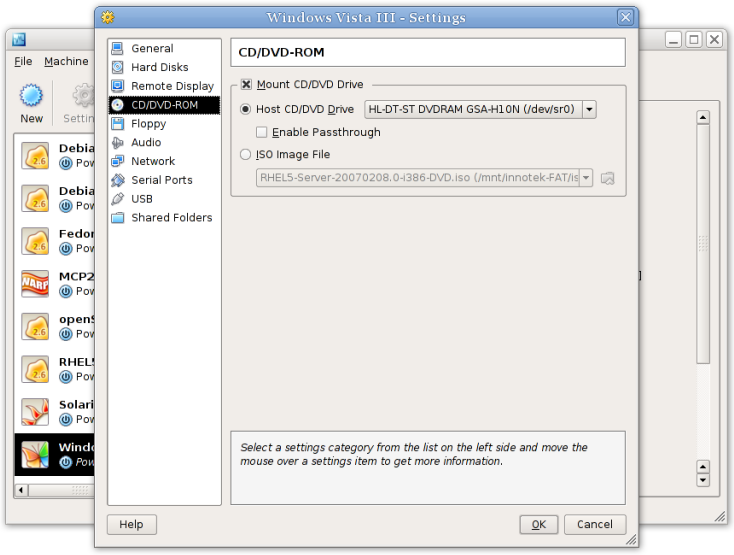
For now, if you have just created an empty VM, you will probably be most interested in the settings presented by the "CD/DVD-ROM" section if want to make a CD-ROM or a DVD-ROM available the first time you start it, in order to install your guest operating system.

For this, you have two options:

- If you have actual CD or DVD media from which you want to install your guest operating system (e.g. in the case of a Windows installation CD or DVD), put the media into your host's CD or DVD drive.

  Then, in the settings dialog, go to the "CD/DVD-ROM" section and select "Host drive" with the correct drive letter (or, in the case of a Linux host, 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 (most probably in the case of a Linux distribution), you would normally burn this file to an empty CD or DVD and proceed as just described. With VirtualBox however, you can skip this step and mount the ISO file directly. 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, in the settings dialog, go to the "CD/DVD-ROM" section and select "ISO image file". This brings up the Virtual Disk Image Manager, where you perform the following steps:

  1. Press the "Add" button to add your ISO file to the list of registered images. This will present an ordinary file dialog that allows you to find your ISO file on your host machine.
  2. Back to the manager window, select the ISO file that you just added and press the "Select" button. This selects the ISO file for your VM.

  The Virtual Disk Image Manager is described in detail in chapter 3.5, *The Virtual Disk Manager*, page 39.

3.4 Running a virtual machine

The "Start" button in the main window starts the virtual machine that is currently selected.

This opens up 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, as can be seen with the image in chapter 1.1, *Virtualization basics*, page 8.

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

3.4.1 Keyboard and mouse support in virtual machines

3.4.1.1 Capturing and releasing keyboard and mouse

Since the operating system 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. This is, however, not the case since, 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.

As a result, initially after installing a host operating system and before you install the guest additions (we will explain this in a minute), only one of the two – your VM or the rest of your computer – can "own" the keyboard and the mouse. You will see a *second* mouse pointer which will always be confined to the limits of the VM window. Basically, you activate the VM by clicking inside it.

To return ownership of keyboard and mouse to your host operating system, VirtualBox reserves a special key on your keyboard for itself: the **"host key".** By default, this is the *right Control key*[1] on your keyboard, but you can change this default in the VirtualBox Global Settings. In any case, the current setting for the host key is always displayed *at the bottom right of your VM window,* should you have forgotten about it:

In detail, all this translates into the following:

- Your **keyboard** is owned by the VM if the VM window on your host desktop has the keyboard focus (and then, if you have many windows open in your guest operating system as well, the window that has the focus in your VM). This means

[1] Actually, the *left Command key* is the default on Mac OS X.

that if you want to type within your VM, click on the title bar of your VM window first.

To release keyboard ownership, press the Host key (as explained above, typically the right Control key).

Note that while the VM owns the keyboard, some key sequences (like Alt-Tab for example) 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 like Alt-Tab will no longer reach the guest.

- 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 type into the VM window, your mouse is not necessarily owned by the VM yet.

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

As this behavior can be inconvenient, VirtualBox provides a set of tools and device drivers for guest systems called the "VirtualBox Guest Additions" which make VM keyboard and mouse operation a lot more seamless. Most importantly, the Additions will get rid of the second "guest" mouse pointer and make your host mouse pointer work directly in the guest.

This will be described later in chapter 4, *The VirtualBox Guest Additions*, page 52.

3.4.1.2 Typing special characters

Operating systems expect certain key combinations to initiate certain procedures. Some of these key combinations may be difficult to enter into a virtual machine, as there are three candidates as to who receives keyboard input: the host operating system, VirtualBox, or the guest operating system. Who of these three receives keypresses depends on a number of factors, including the key itself.

- Host operating systems reserve certain key combinations for themselves. For example, it is impossible to enter the **Ctrl+Alt+Delete** combination if you want to reboot the guest operating system in your virtual machine, because this key combination is usually hard-wired into the host OS (both Windows and Linux intercept this), and pressing this key combination will therefore reboot your *host*.

  Also, with Linux, the key combination **Ctrl+Alt+Backspace** normally resets the X server (to restart the entire graphical user interface in case it got stuck). As the X server intercepts this combination, pressing it will usually restart your *host* graphical user interface (and kill all running programs, including VirtualBox, in the process).

Third, also with Linux, the key combination Ctrl+Alt+Fx (where Fx is one of the function keys from F1 to F12) normally allows to switch between virtual terminals. As with Ctrl+Alt+Delete, these combinations are intercepted by the host operating system and therefore always switch terminals on the *host*.

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

- Use the items in the "VM" menu of the virtual machine window. There you will find "Insert Ctrl+Alt+Delete" and "Ctrl+Alt+Backspace"; the latter will only have an effect with Linux guests, however.
- Press special key combinations with the Host key (normally the right Control key), which VirtualBox will then translate for the virtual machine:
  - **Host key + Del** to send Ctrl+Alt+Del (to reboot the guest);
  - **Host key + Backspace** to send Ctrl+Alt+Backspace (to restart the graphical user interface of a Linux guest);
  - **Host key + F1** (or other function keys) to simulate Ctrl+Alt+F1 (or other function keys, i.e. to switch between virtual terminals in a Linux guest).

- For some other keyboard combinations such as **Alt-Tab** (to switch between open windows), VirtualBox allows 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" -> "Global settings" -> "Input" -> "Auto-capture keyboard".

3.4.2 Changing removable media

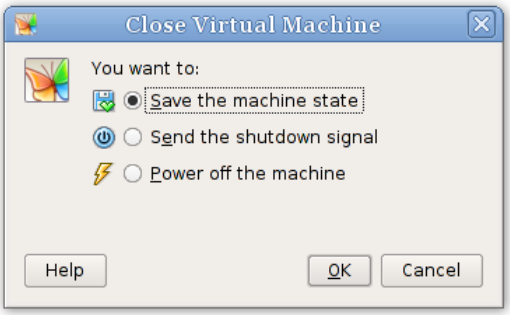
While a virtual machine is running, you can change removable media from the "Devices" menu of the VM's window. Here you can select in detail what VirtualBox presents to your VM as a CD, DVD, or floppy.

The settings are the same as would be available for the VM in the "Settings" dialog of the VirtualBox main window, but since that dialog is disabled while the VM is in "running" or "saved" state, this extra menu saves you from having to shut down and restart the VM every time you want to change media.

Hence, in the "Devices" menu, VirtualBox allows you to attach the host drive to the guest or select a floppy or DVD image using the Disk Image Manager, all as described in chapter 3.3, *Basics of virtual machine configuration*, page 32.

3.4.3 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) (or press the Host key together with "Q"), VirtualBox asks you whether you want to "save" or "power off" the VM.



The difference between these two options is crucial. They mean:

- **Save the machine state:** With this option, VirtualBox "freezes" the virtual machine by completely saving its state to your local disk. When you later resume the VM (by again clicking the "Start" button in the VirtualBox main window), 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 similar to suspending a laptop computer (e.g. 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. So long as a a fairly modern operating system is installed and running in the VM, this should trigger a proper shutdown mechanism in the VM.

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

  This is equivalent of pulling the power plug on a real computer without shutting it down properly. If you start the machine again after powering it off, your operating system 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.

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

3.4.4 Snapshots

With VirtualBox's 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.

This is particularly useful for making sure that a guest installation is not damaged by accidental changes, misbehaving software, or viruses.

Once you have set up the machine the way you want it, simply take a snapshot, and should anything happen to the installation, you can simply revert to its snapshot state.

To **take a snapshot** of your VM, perform the following steps:

1. If your VM is currently in either the "saved" or the "powered off" state (as displayed next to the VM in the VirtualBox main window), click on the "Snapshots" tab on the top right of the main window, and then on the small camera icon (for "Take snapshot").

   If your VM is currently running, select "Take snapshot" from the "VM" pull-down menu of the VM window.

2. A window will pop up and ask you to name the snapshot. 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 external drivers".

3. Your new snapshot will then appear in the list of snapshots under the "Snapshots" tab. Underneath, 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.

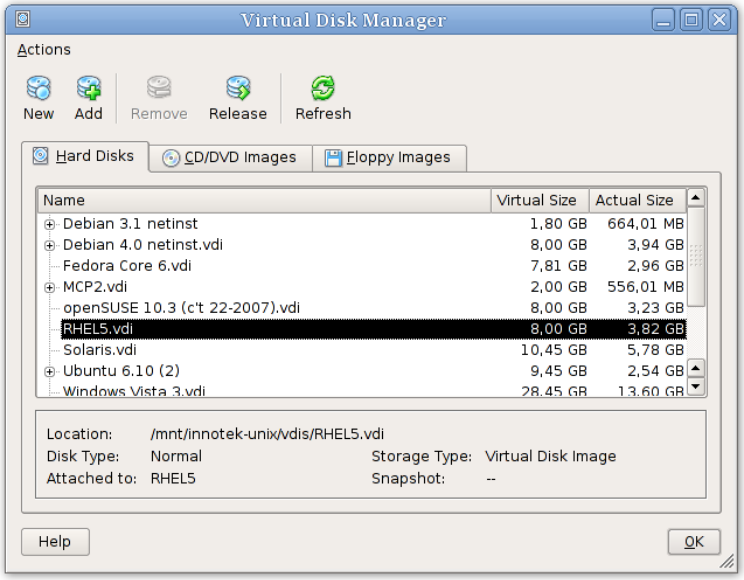
   (If you later take another snapshot, you will see that they will be displayed in sequence, and each subsequent snapshot is a derivation of the earlier one.)

To **revert to an earlier snapshot,** you click on the "Current state" item and select "Discard current state". This will bring the VM back to the state of the nearest (most recent) snapshot. In the same way, you can merge several earlier snapshots into one.

Note: The snapshot reverted to 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 chapter 5, *Virtual storage*, page 62 for details.

3.5 The Virtual Disk Manager

VirtualBox keeps an internal registry of all available hard disk, CD/DVD-ROM and floppy disk images. This registry can be viewed and changed in the **Virtual Disk Manager**, which you can access from the "File" menu in the VirtualBox main window:



The Disk Image Manager shows you all images that are currently registered with VirtualBox, conveniently grouped in three tabs for the three possible formats. These formats are:

- Hard disk images, either in VirtualBox's own Virtual Disk Image (VDI) format or in the widely supported VMDK format;
- CD/DVD images in standard ISO format;
- floppy images in standard RAW format.

Starting with version 1.4, VirtualBox also supports the widely supported VMDK format. This means that if you have created virtual hard disks with another virtualization product that uses the VMDK format, you will not have to recreate these images with VirtualBox, but can continue to use them. See chapter 5.4, *VMDK image files*, page 66 for details.

As you can see in the screenshot above, for each image, the Virtual Disk Manager shows you the full path of the image file and other information, such as the virtual machine the image is currently attached to, if any. Also, as can be seen in the screenshot, if you have created snapshots for a virtual machine, additional "differencing" hard disk images may automatically be created; see chapter 3.4.4, *Snapshots*, page 38 for details.

The Virtual Disk Manager allows you to

- create new hard disk images using the **"New"** button; this will bring up the "Create Disk Image" wizard already described in chapter 3.2, *Creating a virtual machine*, page 28;

- import existing VDI or VMDK files from your hard drive into VirtualBox using the **"Add"** button;
- **remove** an image from the registry (and optionally delete the image file when doing so);
- **"release"** an image, that is, detach it from a virtual machine if it is currently attached to one as a virtual hard disk.

We recommend that you maintain two special folders on your system for keeping images: one for hard disk image files (which can, in the case of dynamically expanding images, grow to considerable sizes), and one for ISO files (which were probably downloaded from the Internet).

Hard disk image files can be copied onto other host systems and imported into virtual machines there, although certain guest systems (notably Windows 2000 and XP) will require that the new virtual machine be set up 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 virtual machine, VirtualBox will complain with an error, since VirtualBox assigns a unique identifier (UUID) to each disk image to make sure it is only used once. See chapter 5.3, *Cloning disk images*, page 65 for instructions on this matter.

Details about the different container formats supported by VirtualBox are described in chapter 5, *Virtual storage*, page 62.

3.6 Deleting virtual machines

The "Delete" button in the main VirtualBox window lets you remove a virtual machine which you no longer need. All settings for that machine will be lost. However, any hard disk images attached to the machine will be kept; you can delete those separately using the Disk Image Manager (described just above).

You cannot delete a machine which has snapshots or is in a saved state, so you must discard these first.

3.7 Virtual machine settings

Most of the settings described below are available in the settings window after selecting a virtual machine in the VirtualBox main window and clicking on the "Settings" button. To keep the user interface simple, those of the following settings which are not as commonly used are not shown in that settings window. They are, however, available through `VBoxManage` and will be described in chapter 8, *VBoxManage reference*, page 96 later.

3.7.1 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. There are four tabs, "Basic", "Advanced", "Description" and "Other".

3.7.1.1 "Basic" tab

Under the "Basic" tab of the "General" settings category, you can find these settings:

Name The name under which the VM is shown in the list of VMs in the main window. Under this name, VirtualBox also saves the VM's configuration files. By changing the name, VirtualBox renames these files as well. As a result, you can only use characters which are allowed in your host operating system's file names.

Note that internally, VirtualBox uses unique identifiers (UUIDs) to identify virtual machines. You can display these with `VBoxManage`.

OS Type The type of the guest operating system that is (or will be) installed in the VM. This is the same setting that was specified in the "New Virtual Machine" wizard, as described with chapter 3.2, *Creating a virtual machine*, page 28 above.

Base Memory size (RAM) The amount of RAM that is allocated and given to the VM when it is running. The specified amount of memory will be allocated from the host operating system (from resident memory 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). Again, this is the same setting that was specified in the "New Virtual Machine" wizard, as described with guidelines under chapter 3.2, *Creating a virtual machine*, page 28 above.

Generally, it is possible to change the memory size after installing the guest operating system (provided you do not reduce the memory to an amount where the operating system would no longer boot).

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

Video memory size 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, but for most setups, the default video memory size of 8MB should be sufficient.

3.7.1.2 "Advanced" tab

Boot order This setting determines the order in which the guest operating system will attempt to boot from the various virtual boot devices. Analogous to a real PC's BIOS setting, 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), or none of these.

If you select "Network", the VM will attempt to boot from a network. This needs to be configured in detail on the command line; please see chapter 8.5, *VBoxManage modifyvm*, page 102.

Enable ACPI VirtualBox can present the Advanced Configuration and Power Interface (ACPI) to the guest operating system for configuring the virtual hardware. In addition, via ACPI, VirtualBox can present the host's power status information to the guest.

ACPI is the current industry standard to allow operating systems to recognize hardware, configure motherboards and other devices and manage power. As all modern PCs contain this feature and Windows and Linux have been supporting it for years, it is also enabled by default in VirtualBox.

> **Warning:** All Windows operating systems starting with Windows 2000 install different kernels depending on whether ACPI is available, so ACPI *must not be turned off* after installation. Turning it on after installation will have no effect however.

Enable I/O APIC Advanced Programmable Interrupt Controllers (APICs) are a newer x86 hardware feature that have replaced old-style Programmable Interrupt Controllers (PICs) in recent years. With an I/O APIC, operating systems can use more than 16 interrupt requests (IRQs) and therefore avoid IRQ sharing for improved reliability.

However, software support for I/O APICs has been unreliable with some operating systems 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 operating systems starting with Windows 2000 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 VT-x/AMD-V This setting determines whether the virtualization engine will try to use the host CPU's hardware virtualization extensions such as Intel VT-x and AMD-V. Normally, you should leave this setting disabled as VirtualBox

employs sophisticated software techniques which normally yield superior performance compared to hardware virtualization. However, for some rather exotic guest operating systems such as OS/2 this setting needs to be enabled.

This is a tri-state setting: aside from enabled and disabled, the checkbox can be in a gray state, which means that the value of the setting shall be determined according to the VirtualBox global preferences (accessible from the "File" menu).

Note: Starting with VirtualBox 1.6, for stability reasons, all *running* virtual machines must have the same setting with respect to hardware virtualization. In other words, if you first start a machine that has hardware virtualization enabled and then start one that has it disabled, you will get a warning message, and the setting of the second machine that was started (and all further machines) will be ignored.

Enable PAE/NX This setting 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 operating system, 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 operating systems (such as Ubuntu Server) require PAE support from the CPU and cannot be run in a virtual machine without it. However, enabling this setting currently has no effect on how much memory can be assigned to the virtual machine.

Shared clipboard If the virtual machine has Guest Additions installed, you can select here whether the clipboard of the guest operating system should be shared with that of your host. If you select "Bidirectional", then VirtualBox will always make sure that both clipboards contain the same data. If you select "Host to guest" or "Guest to host", then VirtualBox will only ever copy clipboard data in one direction.

Snapshot folder By default, VirtualBox saves snapshot data together with your other VirtualBox configuration data; see chapter 9.1, *VirtualBox configuration data*, page 114. With this setting, you can specify any other folder for each VM.

3.7.1.3 "Description" tab

Here you can enter any description for your virtual machine, if you want. This has no effect of the functionality of the machine, but you may find this space useful to note down things like the configuration of a virtual machine and the software that has been installed into it.

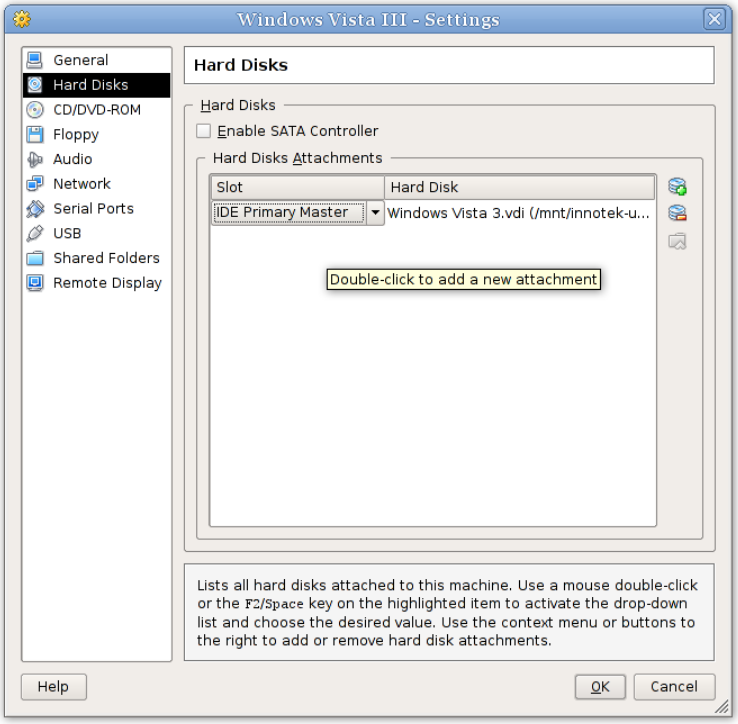
3.7.1.4 Other settings

Remember mounted media at runtime If this is checked, VirtualBox will save the state of what media has been mounted between several runs of a virtual machine.

BIOS logo customization By default, when the virtual machine starts up, VirtualBox displays a VirtualBox logo. With `VBoxManage`, you can change this logo to one of your choice. This setting can only be customized via `VBoxManage`; see chapter 8.5, *VBoxManage modifyvm*, page 102.

3.7.2 Hard disks

In the VM Settings window, the "Hard Disks" section allows you to connect virtual hard disk images to your virtual machine:



As with a real PC, VirtualBox by default offers you two IDE controllers, each with a "master" and a "slave" connection. With one of these four connectors being reserved to the CD-ROM/DVD drive (see below), that leaves you with three possible hard disks, each represented by one disk image file.

The settings of the first disk ("Primary Master") are initially set by the "Create VM" wizard. Normally, you will stick with this setting for the rest of a VM's lifetime. You may, however, freely remove, add and exchange virtual hard drives after the machine

has been set up. 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.

To connect an additional disk, double-click on the empty space in the list of virtual disks, or click on the respective icon to the right of that list. You can then select where the virtual disk should be connected to (primary master or slave or secondary slave) and which image to use. If you click on the "Select virtual disk" icon to the right, this will bring up the Virtual Disk Image Manager (see chapter 3.5, *The Virtual Disk Manager*, page 39 for details), where you can select a different image.

To remove a virtual disk, select it and click on the "remove" icon on the right.

In addition to the IDE controller, VirtualBox can also present an SATA controller to the guest; however, this may require that you run a modern guest operating system. See chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62 for details.

We have dedicated an entire chapter of this User Manual to virtual storage: please see chapter 5, *Virtual storage*, page 62.

3.7.3 CD/DVD-ROM and floppy settings

In the VM Settings window, the settings in these two categories determine what VirtualBox provides as a floppy disk and as a CD/DVD-ROM drive to your VM's guest operating system. For both the floppy and CD/DVD-ROM categories, the following options are available:

- **Not mounted:** The virtual device is presented as empty, that is, no floppy (or no CD/DVD-ROM) is present.
- **Host drive:** The physical device of the host computer is connected to the VM, so that the guest operating system 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.
- **Image file:** Quite similar to virtual hard disks, this presents a file on your host as a device to the guest operating system. To use an image file, you must first import it into the Virtual Disk Manager; see chapter 3.5, *The Virtual Disk Manager*, page 39. The image file format varies depending on the type of device:
  - For floppies, the file must be in raw format.
  - For CD- and DVD-ROMs, the file must be in ISO format. Most commonly, you will select this option when installing an operating system from an ISO file that you have obtained from the Internet. For example, most Linux distributions are available in this way.

All these settings 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.

Note: The identification string of the drive provided to the guest (which, in the guest, would be 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 operating system every time the configuration is changed.

Using the host drive normally provides a read-only drive to the guest. As an experimental feature (which currently works for data only, audio is not supported), it is possible to give the guest access to the CD/DVD writing features of the host drive (if available):

```
VBoxManage modifyvm <vmname> -dvdpassthrough on
```

See also chapter 8.5, *VBoxManage modifyvm*, page 102.

3.7.4 Audio settings

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

If audio is enabled for a guest, you can choose between the emulation of an Intel AC'97 controller or a SoundBlaster 16 card. In any case, you can select what audio driver VirtualBox will use on the host. On Linux hosts, you can select between the OSS, ALSA or the PulseAudio subsystem. On newer Linux distributions (Fedora8+, Ubuntu 8.04+) the PulseAudio subsystem should be preferred.

3.7.5 Network settings

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

When you first create a virtual machine, VirtualBox by default enables one of these four cards and selects the "Network Address Translation" (NAT) mode for it. This way the 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.

Note: If you are installing Windows Vista in a virtual machine, you will probably have no networking initially. See chapter 4.2.4, *Windows Vista networking*, page 55 for instructions how to solve this problem.

In most cases, the "NAT" setting will work fine for you. However, since VirtualBox is extremely flexible in how it can virtualize networking, we have dedicated an entire chapter of this manual to discussing networking configuration; please see chapter 6, *Virtual networking*, page 69.

3.7.6 USB support

3.7.6.1 USB settings

The "USB" section in a virtual machine's Settings window allows you to configure VirtualBox's sophisticated USB support.

VirtualBox can allow virtual machines to access the USB devices on your host directly. To achieve this, VirtualBox presents to the guest operating system 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.

In addition to allowing a guest access to your local USB devices, VirtualBox even allows your guests to connect to remote USB devices by use of the VRDP protocol. For details about this, see chapter 7.4.3, *Remote USB*, page 93.

In the Settings dialog, you can first configure whether USB is available in the guest at all, and in addition also optionally enable the USB 2.0 (EHCI) controller for the guest. If so, you can determine in detail which devices are available. For this, you must create so-called "filters" by specifying certain properties of the USB device.

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 referencing it later) 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:

1. **Vendor and product ID.** With USB, each vendor of USB products carries an identification number that is unique world-wide, the "vendor ID". Similarly, each line of products is assigned a "product ID" number. Both numbers are commonly written in hexadecimal (that is, they are composed of the numbers 0-9 and the letters A-F), 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 and product IDs, you can use the following command (see chapter 8, *VBoxManage reference*, page 96):

```
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.

2. **Serial number.** While vendor and product ID are already 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.

3. **Remote.** This setting specifies whether the device will be local only, or remote only (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 (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 clicking in the checkbox next to the filter name.

3.7.6.2 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 allows VirtualBox to capture devices when they are plugged in, and a USB device driver to claim USB devices for a particular virtual machine. As opposed to VirtualBox versions before 1.4.0, system reboots are no longer necessary after installing the driver. Also, you no longer need to replug devices for VirtualBox to claim them.

On Linux hosts, VirtualBox accesses USB devices on Linux through the `usbfs` file system. Therefore, the user executing VirtualBox needs read and write permission to the USB file system. Most distributions provide a group (e.g. `usbusers`) which the VirtualBox user needs to be added to. Also, VirtualBox can only proxy to virtual machines USB devices which are not claimed by a Linux host USB driver. Please refer to the `Driver=` entry in `/proc/bus/usb/devices` to see which devices are claimed.

3.7.7 Remote display

In the "Remote display" section of a virtual machine's settings, you can enable the VRDP server that is built into VirtualBox to allow you to connect to the virtual machine remotely. For this, you can use any standard RDP viewer, such as the one that comes with Microsoft Windows (typically found under "Accessories" -> "Communication" -> "Remote Desktop Connection") or, on Linux system, the standard open-source `rdesktop` program.

These features are described in detail in chapter 7.4, *Remote virtual machines (VRDP support)*, page 90.

3.7.8 Shared folders

Shared folders allow you to easily exchange data between a virtual machine and your host. This feature requires that the VirtualBox Guest Additions be installed in a virtual machine and are described in detail in chapter 4.5, *Folder sharing*, page 59.

3.7.9 Serial ports

Starting with version 1.5, VirtualBox fully supports virtual serial ports in a virtual machine in an easy-to-use manner.

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). While these are no longer as important as they were until a few years ago (especially since mice are no longer connected to serial ports these days), 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. In other words, 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 operating system sees it a standard 16450-type serial port. Both receiving and transmitting data is supported. How this virtual serial port is then connected to the host is configurable, and details depend on your host operating system.

You can use either the graphical user interface or the command-line `VBoxManage` tool to set up virtual serial ports. For the latter, please refer to chapter 8.5, *VBoxManage modifyvm*, page 102; in that section, look for the `-uart` and `-uartmode` options.

In either case, you can configure up to two virtual serial ports simultaneously. For each such device, you will need to determine

1. what kind of serial port the virtual machine should see by selecting an I/O base address and interrupt (IRQ). For these, we recommend to use the traditional values[2], which are:

   a) COM1: I/O base 0x3F8, IRQ 4

   b) COM2: I/O base 0x2F8, IRQ 3

   c) COM3: I/O base 0x3E8, IRQ 4

   d) COM4: I/O base 0x2E8, IRQ 3

[2] See, for example, http://en.wikipedia.org/wiki/COM_(hardware_interface).

2. Then, you will need to determine what this virtual port should be connected to. For each virtual serial port, you have the following options:

   - You can elect to have the virtual serial port "disconnected", which means that the guest will see it as hardware, but it will behave as if no cable had been connected to it.
   - You can connect 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 OpenSolaris hosts, it will be a device node like `/dev/ttyS0`). VirtualBox will then simply redirect all data received from and sent to the virtual serial port to the physical device.
   - You can tell VirtualBox to connect the virtual serial port to a software pipe on the host. This depends on your host operating system:
     - On a Windows host, data will be sent and received through a named pipe. You can use a helper program called VMWare Serial Line Gateway, available for download at `http://www.l4ka.org/tools/vmwaregateway.php`. This tool provides a fixed server mode named pipe at `\\.\pipe\vmwaredebug` and connects incoming TCP connections on port 567 with the named pipe.
     - On a Mac, Linux or OpenSolaris host, a local domain socket is used instead. 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 VirtualBox should create the named pipe (or, on non-Windows hosts, the local domain socket) itself or whether VirtualBox should assume that the pipe (or socket) exists already. With the `VBoxManage` command-line options, this is referred to as "server" or "client" mode, respectively.

Up to two serial ports can be configured simultaneously per virtual machine, but you can pick any port numbers out of the above. For example, you can configure two serial ports to be able to work with COM2 and COM4 in the guest.

4 The VirtualBox Guest Additions

The previous chapter covered getting started with VirtualBox and installing operating systems in a virtual machine. For any serious and interactive use, the 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

As said in chapter 1.1, *Virtualization basics*, page 8, the Guest Additions are designed to be installed *inside* a virtual machine. They consist of device drivers and system applications for the guest operating system that optimize the guest for better performance and usability. To install these additions, you simply provide a special ISO file that comes with VirtualBox as a virtual CD-ROM to your guest operating system and install from there.

Please see chapter 1.3.2, *Supported guest operating systems*, page 13 for details on what guest operating systems are fully supported with Guest Additions by VirtualBox. The Guest Additions offer the following features:

Mouse pointer integration To overcome the limitations for mouse support that were described in chapter 3.4.1.1, *Capturing and releasing keyboard and mouse*, page 35, this provides you with seamless mouse support. Essentially, a special mouse driver is installed in the guest that communicates with the "real" mouse driver on your host and moves the guest mouse pointer accordingly. 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.

Better video support While the virtual graphics card which 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 and recent Linux, Solaris and OpenSolaris guests, when the Guest Additions are installed, you can resize the virtual machine's window, and 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).

Time synchronization With the Guest Additions installed, VirtualBox can ensure that the guest's system time is better synchronized. This fixes the problem that an operating system normally expects to have 100% of a computer's time for itself

without interference, which is no longer the case when your VM runs together with your host operating system and possibly other applications on your host. As a result, your guest operating system's timing will soon be off significantly. The Guest Additions will re-synchronize the time regularly.

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 VirtualBox to treat a certain folder as a shared folder, and VirtualBox will make it available to the guest operating system as a network share. For details, please refer to chapter 4.5, *Folder sharing*, page 59.

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 chapter 4.6, *Seamless windows*, page 60 for details.

Shared clipboard With the Guest Additions installed, the clipboard of the guest operating system can optionally be shared with your host operating system; see chapter 3.7.1, *General settings*, page 42.

Automated Windows logons (credentials passing; Windows guests only). For details, please see chapter 9.2, *Automated Windows guest logons (VBoxGINA)*, page 115.

4.2 Windows Guest Additions

The 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 Vista (all editions)

Generally, it is strongly recommend to install the Windows Guest Additions.

4.2.1 Installing the Windows Guest Additions

The VirtualBox Guest Additions are provided as a CD-ROM image file which is called `VBoxGuestAdditions.iso`. An easy-to-use installation program will guide you through the setup process. As VirtualBox can provide ISO files as virtual CD-ROM drives to the Windows guests, Windows can automatically install these additions.

4.2.1.1 Mounting the Additions ISO file

In the "Devices" menu in the virtual machine's menu bar, VirtualBox has a handy menu item named "Install guest additions", which will automatically bring up the Additions in your VM window.

If you prefer to mount the 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 brings up the Virtual Disk Manager described in chapter 3.5, *The Virtual Disk Manager*, page 39.

3. In the Virtual Disk Manager, press the "Add" button and browse your host file system for the `VBoxGuestAdditions.iso` file:

   - On a Windows host, you can find this file in the VirtualBox installation directory (usually under `C:\Program files\Sun\xVM VirtualBox`).
   - On a Linux host, you can find this file in the `additions` folder under where you installed VirtualBox (normally `/opt/VirtualBox-1.6.2`).
   - On OpenSolaris hosts, you can find this file in the `additions` folder under where you installed VirtualBox (normally `/opt/VirtualBox`).
   - On Mac OS X hosts, you can find this file in the application bundle of VirtualBox. (Right click on the VirtualBox icon in Finder and choose *Show Package Contents*. There it is located in the `Contents/MacOS` folder.)

4. Back in the Virtual Disk Manager, select that ISO file and press the "Select" button. This will mount the ISO file and present it to your Windows guest as a CD-ROM.

4.2.1.2 Running the installer

Unless you have the Autostart feature disabled in your Windows guest, Windows will now autostart the VirtualBox Guest Additions installation program from the Additions ISO. If the Autostart feature has been turned off, choose `setup.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.

4.2.2 Updating the Windows Guest Additions

Windows Guest Additions can be updated by running the installation program again, as previously described. This will then replace the previous Additions drivers with updated versions.

Alternatively, you may also open the Windows Device Manager and select "Update driver..." for two devices:

1. the VirtualBox Graphics Adapter and
2. the VirtualBox System Device.

For each, choose to provide your own driver and use "Have Disk" to point the wizard to the CD-ROM drive with the Guest Additions.

4.2.3 Unattended Installation

In order to allow for completely unattended guest installations of Windows 2000 and XP, the Guest Additions driver files have been put separately on the Additions ISO file. Just like with other third-party drivers, the files have to be copied to the OEM directory of Windows. Using the PCI hardware detection, they will then be recognized and installed automatically.

4.2.4 Windows Vista networking

Windows Vista no longer ships a driver for the AMD PCnet Ethernet card that VirtualBox provides to the guest by default. As a result, after installation of Vista in a virtual machine, there will be no networking initially. As a convenience, VirtualBox ships with a driver for the AMD PCnet card, which comes with the Windows Guest Additions. If you install these in a Vista guest, the driver will automatically be installed as well.

If, for some reason, you would like to install the driver manually, you can find it on the Guest Additions ISO. To install this driver, mount the Guest Additions ISO (as described above, select "Install guest additions" from the "Devices" menu). Then, start the Windows Hardware Wizard and direct it to the Guest Additions CD where a driver for the PCnet card can be found in the directory `AMD_PCnet`.

Alternatively, change the Vista guest's VM settings to use an Intel networking card instead of the default AMD PCnet card; see chapter 3.7.5, *Network settings*, page 47 for details.

4.3 Linux Guest Additions

Like the Windows Guest Additions, the VirtualBox Guest Additions for Linux take the form of a set of device drivers and system applications which may be installed in the guest operating system.

The following Linux distributions are officially supported:

- Fedora Core 4, 5, 6, 7 and 8
- Redhat Enterprise Linux 3, 4 and 5
- SUSE and openSUSE Linux 9, 10.0, 10.1, 10.2 and 10.3
- Ubuntu 5.10, 6.06, 7.04, 7.10 and 8.04

Other distributions may work if they are based on comparable software releases.

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.

As with Windows guests, we recommend installation of the VirtualBox Guest Additions for Linux.

4.3.1 Installing the Linux Guest Additions

The VirtualBox Guest Additions for Linux are provided on the same ISO CD-ROM as the Additions for Windows described above. They also come with an installation program guiding you through the setup process, although, due to the significant differences between Linux distributions, installation may be slightly more complex.

Installation involves the following steps:

1. Before installing the Guest Additions, you will have to prepare your guest system for building external kernel modules. This is exactly the same process as described in chapter 2.3.2, *The VirtualBox kernel module*, page 17, except that this step must now be performed in your Linux *guest* instead of on a Linux host system, as described there.

2. Mount the `VBoxGuestAdditions.iso` file as your Linux guest's virtual CD-ROM drive, exactly the same way as described for a Windows guest in chapter 4.2.1.1, *Mounting the Additions ISO file*, page 54.

3. Change to the directory where your CD-ROM drive is mounted and execute as root:

   ```
   sh ./VBoxLinuxAdditions.run
   ```

The VirtualBox Guest Additions contain several different drivers. If for any reason you do not wish to install them all, you can specify the ones which you wish on the command line - for example

```
sh ./VBoxAdditions.run x11
```

to install the X Window graphic drivers. Type in the command

```
sh ./VBoxAdditions.run help
```

for more information.

4.3.2 Video acceleration and high resolution graphics modes

In Linux guests, VirtualBox video acceleration is available through the X Window System. Typically, in today's Linux distributions, this will be the X.Org server. During the installation process, X will be set up to use the VirtualBox video driver. On recent Linux guests (that is, guests running X.Org server version 1.3 or later) graphics modes can be selected by resizing the VirtualBox window using the mouse, or sending video mode hints using the VBoxManage tool. If you are only using recent Linux guests systems, you can skip the rest of this section. On older guest systems, whatever graphics modes were set up before the installation will be used. If these modes do not suit your requirements, you can change your setup by editing the configuration file of the X server, usually found in `/etc/X11/xorg.conf`.

VirtualBox can use any default X graphics mode which fits into the virtual video memory allocated to the virtual machine, as described in chapter 3.7.1, *General settings*, page 42. You can also add your own modes to the X server configuration file. You simply need to add them to the "Modes" list in the "Display" subsection of the "Screen" section. For example, the section shown here has a custom 2048x800 resolution mode added:

```
Section "Screen"
        Identifier    "Default Screen"
        Device        "VirtualBox graphics card"
        Monitor       "Generic Monitor"
        DefaultDepth  24
        SubSection "Display"
                Depth         24
                Modes         "2048x800" "800x600" "640x480"
        EndSubSection
EndSection
```

4.3.3 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.

4.4 Solaris Guest Additions

Like the Windows Guest Additions, the VirtualBox Guest Additions for Solaris take the form of a set of device drivers and system applications which may be installed in the guest operating system.

The following Solaris distributions are officially supported:

- OpenSolaris Nevada (Build 82 and higher)
- OpenSolaris Indiana (Developer Preview 2 and higher)
- Solaris 10 (u5 and higher)

Other distributions may work if they are based on comparable software releases.

As with Windows and Linux guests, we recommend installation of the VirtualBox Guest Additions for Solaris.

4.4.1 Installing the Solaris Guest Additions

The VirtualBox Guest Additions for Solaris are provided on the same ISO CD-ROM as the Additions for Windows and Linux described above. They also come with an installation program guiding you through the setup process.

Installation involves the following steps:

1. Mount the `VBoxGuestAdditions.iso` file as your Solaris guest's virtual CD-ROM drive, exactly the same way as described for a Windows guest in chapter 4.2.1.1, *Mounting the Additions ISO file*, page 54.

   If in case the CD-ROM drive on the guest doesn't get mounted (observed on some versions of Solaris 10), execute as root:

   ```
   svcadm restart volfs
   ```

2. Change to the directory where your CD-ROM drive is mounted and execute as root:

   ```
   pkgadd -d ./VBoxSolarisAdditions.pkg
   ```

3. Choose "1" and confirm installation of the guest additions package. After the installation is complete, re-login to X server on your guest to activate the X11 Guest Additions.

Note: Please note that Shared Folders are not yet available for Solaris guests.

4.4.2 Uninstalling the Solaris Guest Additions

The Solaris Guest Additions can be safely removed by removing the package from the guest. Open a root terminal session and execute:

```
pkgrm SUNWvboxguest
```

4.4.3 Updating the 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.

4.5 Folder sharing

Shared folders allow you to access files of your host system from within the guest system, much like ordinary shares on Windows networks would – except that shared folders do not need a networking setup. Shared folders must physically reside on the *host* and are then shared with the guest; sharing is accomplished using a special service on the host and a file system driver for the guest, both of which are provided by VirtualBox.

In order to use this feature, the VirtualBox Guest Additions have to be installed. Note however that Shared Folders are only supported with Windows (2000 or newer) guests and Linux guests.

To share a host folder with a virtual machine in VirtualBox, you must specify the path of that folder and choose for it a "share name" that the guest can use to access it. Hence, first create the shared folder on the host; then, within the guest, connect to it.

There are several ways in which shared folders can be set up for a particular virtual machine:

- In the graphical user interface of a running virtual machine, you can select "Shared folders" from the "Devices" menu, or click on the folder icon on the status bar in the bottom right corner of the virtual machine window.

- If a virtual machine is not currently running, you can configure shared folders in each virtual machine's "Settings" dialog.

- From the command line, you can create shared folders using the the VBoxManage command line interface; see chapter 8, *VBoxManage reference*, page 96. The command is as follows:

```
VBoxManage sharedfolder add "VM name" -name "sharename"
                    -hostpath "C:\test"
```

There are two types of shares:

1. VM shares which are only available to the VM for which they have been defined;

2. transient VM shares, which can be added and removed at runtime and do not persist after a VM has stopped; for these, add the `-transient` option to the above command line.

Shared folders have read/write access to the files at the host path by default. To restrict the guest to have read-only access, create a read-only shared folder. This can either be achieved using the GUI or by appending the parameter `-readonly` when creating the shared folder with VBoxManage.

Then, you can mount the shared folder from inside a VM the same way as you would mount an ordinary network share:

- In a Windows guest, starting with VirtualBox 1.5.0, shared folders are browseable and are therefore visible in Windows Explorer. So, to attach the host's shared folder to your Windows guest, open Windows Explorer and look for it under "My Networking Places" -> "Entire Network" -> "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:

  ```
  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
  ```

  Replace `sharename` with the share name specified with `VBoxManage`, and `mountpoint` with the path where you want the share to be mounted (e.g. `/mnt/share`). The usual mount rules apply, that is, create this directory first if it does not exist yet.

  Beyond the standard options supplied by the `mount` command, the following are available:

  ```
  iocharset CHARSET
  ```

  to set the character set used for I/O operations (utf8 by default) and

  ```
  convertcp CHARSET
  ```

  to specify the character set used for the shared folder name (utf8 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 allow access by normal users (in read/write mode, depending on the settings) even if root has mounted the filesystem.

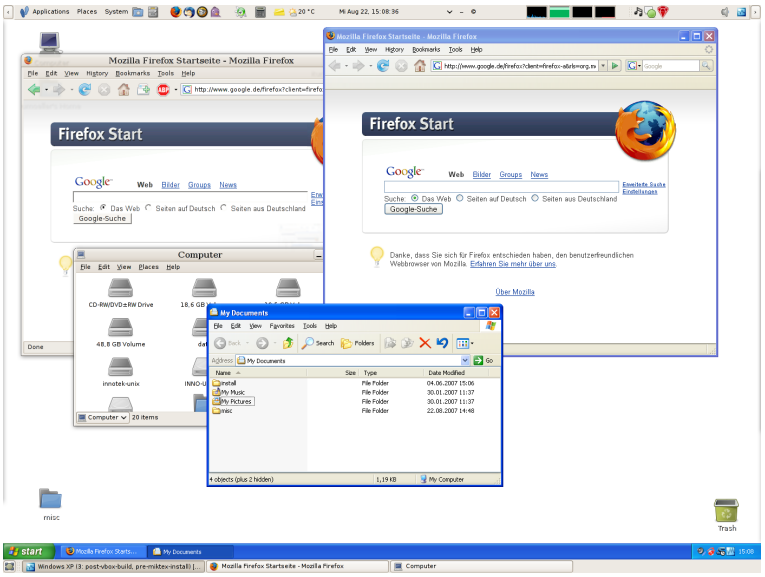
4.6 Seamless windows

With the "seamless windows" feature of 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 (support added with VirtualBox 1.5);

- Linux or Solaris/OpenSolaris guests with an X.org server version 1.3 or higher[1] (support added with VirtualBox 1.6).

After seamless windows are enabled (see below), 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:



To enable seamless mode, after starting the virtual machine, press the Host key (normally the right control key) together with "L". 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 go back to the "normal" VM display (i.e. to disable seamless windows), press the Host key and "L" again.

[1] The X server version differs from the version of the entire X.org suite. You can type `X -version` in a terminal to find out about the X.org server version level that is currently installed.

5 Virtual storage

As the virtual machine will most probably expect to see a hard disk built into its virtual computer, VirtualBox must be able to present "real" storage to the guest as a virtual hard disk. There are presently three methods in which to achieve this:

1. Most commonly, VirtualBox will use large image files on a real hard disk and present them to a guest as a virtual hard disk. This is described in chapter 5.2, *Virtual Disk Image (VDI) files*, page 64.

2. Alternatively, if you have iSCSI storage servers, you can attach such a server to VirtualBox as well; this is described in chapter 5.5, *iSCSI servers*, page 66.

3. Finally, as an experimental feature, you can allow a virtual machine to access one of your host disks directly; this advanced feature is described in chapter 9.9, *Using a raw host hard disk from a guest*, page 121.

Each such virtual storage device (image file, iSCSI target or physical hard disk) will need to be connected to the virtual hard disk controller that VirtualBox presents to a virtual machine. This is explained in the next section.

5.1 Hard disk controllers: IDE, SATA, AHCI

In a real PC, hard disks and CD-ROM/DVD drives are connected to a device called hard disk controller which drives hard disk operation and data transfers. VirtualBox can emulate the two most common types of hard disk controllers typically found in today's PCs: IDE and SATA.[1]

- IDE (ATA) controllers have been in use since the 1980s. Initially, this type of 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 hard disk controllers have two connectors for such cables; as a result, most PCs support up to four devices.

  You can connect up to four virtual storage devices to a virtual machine. Since one of these (the secondary master) is always configured to be a CD-ROM/DVD drive, this leaves you with up to three virtual hard disks that you can attach to a virtual machine's IDE controller.

[1] SATA support was added with VirtualBox 1.6.

- Serial ATA (SATA) is a newer standard introduced in 2003, which supports both higher speeds and more devices per hard disk controller. Also, with real 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).

  The main problem with AHCI controllers is that it is not supported by older operating systems out of the box. Also, for compatibility reasons, AHCI controllers by default operate the disks attached to it in a so-called IDE compatibility mode, unless SATA support is explicitly requested.

  Like a real SATA controller, VirtualBox's virtual SATA controller operates faster and also consumes less CPU resources than the virtual IDE controller. Also, this allows you to connect more than three virtual hard disks to the machine. Finally, a future version of VirtualBox will also implement Native Command Queueing and may thus offer additional performance improvements over IDE.

VirtualBox will always present at least the IDE controller device to the guest. Even if your guest operating system does not support SATA (AHCI), it will always see this IDE controller and the virtual disks attached to it.

If you enable the SATA controller, this will be shown as a separate, additional PCI device to the virtual machine. VirtualBox supports up to 30 SATA slots (numbered 0-29 in the graphical user interface). Of these, the first four (numbered 0-3 in the graphical user interface) are operated in IDE compatibility mode by default.

Warning: The entire SATA controller and the virtual disks attached to it (including those in IDE compatibility mode) will only seen by operating systems that have device support for AHCI. In particular, there is no support for AHCI in Windows before Windows Vista; Windows XP (even SP2) will not see such disks unless you install additional drivers. We therefore do not recommend installing operating systems on SATA disks at this time.

This now gives you the following categories of virtual hard disk slots:

1. three slots attached to the traditional IDE controller, which are always present (plus one for the virtual CD-ROM device);
2. slots attached to the new SATA controller, provided that your guest operating system can see it; these can either be
   a) in IDE compatibility mode (by default, slots 0-3) or
   b) in SATA mode.

To change the IDE compatibility mode settings for the SATA controller, please see chapter 8.5, *VBoxManage modifyvm*, page 102.

5.2 Virtual Disk Image (VDI) files

By default, VirtualBox uses its own container format for guest hard disks – Virtual Disk Image (VDI) files.

The VDI files reside on the host system and are seen by the guest systems as hard disks of a certain geometry. When creating an image, its size has to be specified, which determines this fixed geometry. It is therefore not possible to change the size of the virtual hard disk later.

As briefly mentioned in chapter 3.2, *Creating a virtual machine*, page 28, there are two options of how to create the image: fixed-size or dynamically expanding.

- If you create a **fixed-size image** of e.g. 10 GB, a VDI file of roughly the same size will be created on your host system. 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.

- For more flexible storage management, use a **dynamically expanding image**. This will initially be very small and not occupy any space for unused virtual disk sectors, but the image file will grow every time a disk sector is written to for the first time. While this format takes less space initially, the fact that VirtualBox needs to constantly expand the image file consumes additional computing resources, so until the disk has fully expanded, write operations are slower than with fixed size disks. However, after a dynamic disk has fully expanded, the performance penalty for read and write operations is negligible.

For either of the above two image types (that is, irrespective of whether an image is fixed-size or dynamically expanding), you can also specify whether write operations affect the image directly.

1. With **normal images** (the default setting), there are no restrictions on how guests can read from and write to the disk. Because of this, a normal hard disk can only be attached to a single virtual machine at any given time (although you can detach them from a VM and attach them to another).

   When you take a snapshot of your virtual machine as described in chapter 3.4.4, *Snapshots*, page 38, the state of such a "normal hard disk" will be recorded together with the snapshot, and when reverting to the snapshot, its state will be fully reset.

2. By contrast, **immutable images** are read-only and can be used from multiple virtual machines simultaneously. Write accesses to immutable hard disks will be directed to a special differencing disk image which VirtualBox creates automatically. However, when you shut down the VM to which the immutable disk is attached, the changes in the differencing disk will be completely discarded.

   Clearly, *creating* an immutable virtual disk image makes no sense, because then the hard disk would always be reset to an empty state when its VM is shut down.

Hence, you will ordinarily create a "normal" virtual disk image and then, when its contents are deemed useful, mark it immutable. For reasons of data consistency, it is presently not possible to change the type of a hard disk disk image that is currently registered. As a result, first remove the existing image from VirtualBox's list of registered images using `VBoxManage unregisterimage` and then re-register the image using `VBoxManage registerimage`, adding the `-type immutable` parameter; see chapter 8.10, *VBoxManage registerimage / unregisterimage*, page 109.

3. Finally, **write-through hard disks** are like normal hard disks in that they fully support read and write operations also. However, their state is *not* saved when a snapshot is taken, and not restored when a VM's state is reverted.

   To *create* a disk image as "write-through", use the `VBoxManage createvdi` command; see chapter 8.12, *VBoxManage createvdi*, page 109. To mark an *existing* image as write-through, unregister and re-register the image as previously described, but add `-type writethrough`.

To illustrate the differences between the various types with respect to snapshots: You have installed your guest operating system in your VM, and you have taken a snapshot. Imagine you have accidentally infected your VM with a virus and would like to go back to the snapshot. With a normal hard disk image, you simply revert the state of the VM, 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, irrespective of the snapshot, all it takes is to shut down 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.

Still, you might find write-though images useful if you want to preserve critical data irrespective of snapshots, and since you can attach more than one VDI to a VM, you may want to have one immutable for the operating system and one write-through for your data files.

5.3 Cloning disk images

You can duplicate hard disk image files on the same host to quickly produce a second virtual machine with the same operating system setup. However, you should *only* make copies of virtual disk images using the utility supplied with VirtualBox; see chapter 8.14, *VBoxManage clonevdi*, page 110. This is because VirtualBox assigns a unique identity number (UUID) to each disk image, which is also stored inside the image, and 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 VirtualBox's utility and import that instead.

Note that newer Linux distributions identify the boot hard disk from the ID of the drive. The ID 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 `scsi-SATA_VBOX_HARDDISK_VB5cfdb1e2-c251e503`. The ID for the copied image can be determined with

```
hdparm -i /dev/sda
```

5.4 VMDK image files

Starting with version 1.4, VirtualBox also supports the popular and open VMDK container format that is now supported by a large number of virtualization products.

This means you can import your existing VMDK files by way of the Virtual Disk Manager just like existing VDI images; see chapter 3.5, *The Virtual Disk Manager*, page 39. While VirtualBox fully supports using VMDK files in most situations, the more advanced features of virtual hard disks are presently not supported. In detail, with VMDK images,

- you presently cannot create snapshots;
- only write-through images are supported; immutable and normal hard disk are not.

These restrictions will be overcome in a future release. Creating VMDKs giving raw disk or raw partition access is already implemented; see chapter 9.9, *Using a raw host hard disk from a guest*, page 121.

5.5 iSCSI servers

iSCSI stands for "Internet SCSI" and is a standard that allows for using the SCSI[2] 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 "iSCSI initiator".

VirtualBox can transparently present iSCSI remote storage to a virtual machine as a virtual hard disk. The guest operating system will not see any difference between a virtual disk image (VDI file) and an iSCSI target. To achieve this, VirtualBox has an integrated iSCSI initiator.

[2] SCSI, in turn, is the "Small Computer System Interface" and is an established industry standard for data transfer between devices, notably storage devices. Established as early as 1986, SCSI is still used for connecting hard disks and tape devices even today. Especially in the PC market, however, it competed with other data transfer standards such as IDE. It is still in common use in workstations and servers.

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 VirtualBox, you must first register it as a virtual hard disk with `VBoxManage`; see chapter 8.16, *VBoxManage addiscsidisk*, page 110. The target will show up in the list of disk images, as described in chapter 3.5, *The Virtual Disk Manager*, page 39, and can thus be attached to one of the VM's three hard disk slots the usual way.

Note: As opposed to the VDI files described previously, the type of iSCSI targets cannot be "normal" or "immutable", but will always be set to "write through". This means that their state is not saved or reverted with snapshots.

5.5.1 Access iSCSI targets via Internal Networking

As an experimental feature, VirtualBox allows for accessing an iSCSI target running in a virtual machine which is configured for using Internal Networking mode (as described in chapter 6.9, *Internal networking*, page 85). The setup of the virtual machine which uses such an iSCSI target is done as described above. The only difference is that the IP address of the target must be specified as a numeric IP address.

The IP stack accessing Internal Networking must be configured in the virtual machine which accesses the iSCSI target. A free static IP and a MAC address not used by other virtual machines must be chosen. In the example below, adapt the name of the virtual machine, the MAC address, the IP configuration and the Internal Networking name ('MyIntNet') according to your needs. The following 7 commands must be issued:

```
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/Trusted 1
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/Config/MAC 08:00:27:01:02:0f
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/Config/IP 10.0.99.1
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/Config/Netmask 255.255.255.0
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/LUN#0/Driver IntNet
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/LUN#0/Config/Network MyIntNet
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/IntNetIP/0/LUN#0/Config/IsService 1
```

Finally the iSCSI TCP transport driver needs to be instructed to use the IP stack for Internal Networking instead of the normal host IP stack:

```
VBoxManage setextradata VMNAME
      VBoxInternal/Devices/piix3ide/0/LUN#0/AttachedDriver/AttachedDriver/Config/HostIPStack 0
```

The virtual machine with the iSCSI target should be started before the VM using it is powered on. If a virtual machine using an iSCSI disk is started without having the iSCSI target powered up, it can take up to 200 seconds to detect this situation. The VM will fail to power up.

6 Virtual networking

As briefly mentioned in chapter 3.7.5, *Network settings*, page 47, VirtualBox provides up to four virtual PCI Ethernet cards for each virtual machine. For each such card, you can individually select

1. the hardware that will be virtualized as well as
2. the virtualization mode that the virtual card will be operating in with respect to your physical networking hardware on the host.

6.1 Virtual networking hardware

For each card, you can individually select what kind of *hardware* will be presented to the virtual machine. VirtualBox can virtualize the following four types of networking hardware:

- AMD PCNet PCI II;
- AMD PCNet FAST III (the default);
- Intel PRO/1000 MT Desktop;
- Intel PRO/1000 T Server.

The PCNet FAST III is the default because it is supported by nearly every all operating systems out of the box, as well as the GNU GRUB boot manager. However, support for the Intel PRO/1000 MT type was added with VirtualBox 1.6 because Microsoft dropped support for the AMD PCNet cards with Window Vista, and Vista guests therefore have no networking without manual driver installation otherwise. See chapter 4.2.4, *Windows Vista networking*, page 55 for details. The Server variant of the Intel PRO/1000 card was added with VirtualBox 1.6.2 because this one is recognized by Windows XP guests without additional driver installation.

6.2 Introduction to networking modes

Each of the four networking adapters can be separately configured to operate in one of the following four modes:

- Not attached

- Network Address Translation (NAT)
- Host Interface Networking
- Internal Networking

By default, virtual network cards are set up to use *network address translation*, which is well suited to standard networking needs (accessing the Internet from programs running in the guest and providing network services for machines in a local intranet). In particular, if all you want is to browse the Web, download files and view e-mail inside the guest, then the default configuration of the NAT network should be sufficient for you, and you can safely skip the rest of this section. Please note that the `ping` utility does not work over NAT, and that there are certain limitations when using Windows file sharing.

For advanced networking needs such as network simulations, *host interface networking* can be used to set up an additional, software based network interface on the host to which the virtual machine is connected. Finally, VirtualBox *internal networking* can be used to create a virtual network which is visible to selected virtual machines, but not to applications running on the host or to the outside world. The following sections describe the available network modes in more detail.

6.3 "Not attached" mode

When a virtual network card's mode is set to "Not attached", VirtualBox reports to the guest that a network card is present, but that there is no connection – as if no Ethernet cable was plugged into the card. This way 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.

6.4 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 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 VirtualBox networking engine, which maps traffic from and to the virtual machine transparently. 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 (described below).

The virtual machine receives its network address and configuration on the private network from a DHCP server that is integrated into VirtualBox. The address which the virtual machine receives is usually on a completely different network to 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 for some reason then refer to chapter 9.11, *Configuring the address of a NAT network interface*, page 124.

The network frames sent out by the guest operating system are received by 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 VirtualBox application on the host, using an IP address belonging to the host. VirtualBox listens for replies to the packages sent, and repacks and resends them to the guest machine on its private network.

6.4.1 Configuring port forwarding with NAT

As the virtual machine is connected to a private network internal to 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, VirtualBox can make given services available outside of the guest by using **port forwarding.** This means that VirtualBox listens to certain ports on the host and resends all packages which arrive on them to the guest on the ports used by the services being forwarded. 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 (note that 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 to the host system.

You can set up a guest service which you wish to proxy using the command line tool `VBoxManage`. You will need to know which ports on the guest the service uses and to decide which ports to use on the host (often but not always you will 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. An example of how to set up incoming NAT connections to a `ssh` server on the guest requires the following three commands:

```
VBoxManage setextradata "Linux Guest"
        "VBoxInternal/Devices/pcnet/0/LUN#0/Config/guestssh/Protocol" TCP
VBoxManage setextradata "Linux Guest"
        "VBoxInternal/Devices/pcnet/0/LUN#0/Config/guestssh/GuestPort" 22
VBoxManage setextradata "Linux Guest"
        "VBoxInternal/Devices/pcnet/0/LUN#0/Config/guestssh/HostPort" 2222
```

The name `guestssh` is an arbitrary one chosen for this particular forwarding configuration. With that configuration in place, all TCP connections to port 2222 on the host will be forwarded to port 22 on the guest. `Protocol` can be either of `TCP` or `UDP` (these are case insensitive). To remove a mapping again, use the same commands, but leaving out the values (in this case `TCP`, `22` and `2222`).

It is not possible to configure incoming NAT connections while the VM is running. However you can change the settings for a VM which is currently saved (or powered off at a snapshot).

6.4.2 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`.

6.4.3 NAT limitations

There are four **limitations** of NAT mode which users should be aware of:

ICMP protocol is very limited: Some frequently used network debugging tools (e.g. `ping`) rely on sending/receiving messages based on the ICMP protocol. The ICMP protocol cannot be used directly by normal applications such as VirtualBox, as it would, at least on Linux hosts, require root permissions (more precisely CAP_NET_RAW). Since this is not desirable, no attempt has been made to support ICMP to addresses other than 10.0.2.2 and 10.0.2.15. If you try to ping any other IP address you will not get any response.

Receiving of UDP broadcasts is not reliable: The guest does not reliably receive broadcasts, since, 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 is not always working (but WINS always works). As a workaround, you can use the numeric IP of the desired server in the `\\server\share` notation.

Protocols such as GRE are unsupported: Protocols other than TCP and UDP are not supported. This means some VPN products (e.g. PPTP from Microsoft) cannot be used. There are other VPN products which use simply TCP and UDP.

Forwarding host ports < 1024 impossible: On Unix-based hosts (e.g. Linux, Solaris, MacOS X) it is not possible to bind to ports below 1024 from applications that are not run by `root`. Therefore if you try to configure such a port forwarding, then the VM will refuse to start.

These limitations normally don't 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 (i.e. ports not below 1024).

6.5 Introduction to Host Interface Networking (HIF)

With Host Interface Networking, VirtualBox creates a new networking interface in software on the *host* computer. This new software interface will then exist in parallel to your regular interfaces (e.g., on a Linux host `vbox0` will exist alongside `eth0`). 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.

You can create several VirtualBox host interfaces on the host system (see the following subsections for instructions on how to do so), but each of them can only be connected to a single virtual network card in a single guest at one time. In other words, for each virtual network card that is supposed to use Host Interface Networking, you will need to set up a new interface on the host.

Warning: Setting up Host Interface Networking requires changes to your host's network configuration, which will cause the host to temporarily lose its network connection. Do not change network settings on remote or production systems unless you know what you are doing. In particular, you probably do not want to configure host interface networking for a remote machine which you have connected to via `ssh`.

There are few limits on the number of setups which can be created using Host Interface Networking. For the sake of simplicity, we will only describe a simple setup using network bridging for the different host operating systems that VirtualBox supports. For more advanced networking needs, we recommend that you consult general documentation about networking on your host operating system.

Network bridging is one of the simplest ways to use Host Interface Networking. Bridging allows you to connect several network devices together in software, so that data sent to one of the devices will be sent to all of them. For our purposes, this means that virtual machines can send packages through the host's network card, using their own network hardware address, and receive packages sent to it. Other computers on your network will see your guests as though they were physically connected to the network. You will need wired (Ethernet) network hardware on the host for this as most current wireless network devices do not support bridging.

In some network environments (often company networks), measures are taken to prevent several MAC addresses being used on a single network interface by temporarily blocking communication to that interface. This is intended to prevent certain types of network attacks, but will also prevent bridging setups from working correctly.

6.6 Host Interface Networking and bridging on Windows hosts

When you install VirtualBox on a Windows host, the setup program installs a special networking driver on your system. This driver, the VirtualBox Host Interface NDIS driver, can be used to create additional host interfaces. These must be created explicitly before they can be attached to a virtual machine.

Use the `VBoxManage` tool to create new host interfaces on your Windows system:

```
VBoxManage createhostif "VM1 external"
```

Alternatively you can use the network configuration in the VirtualBox GUI to create and delete host interfaces.

Each new host interface thus created appears as an additional network card in your standard Windows "Network Connections" properties. After you have created your new host interface this way, you can select "Host Interface" as the networking mode in a virtual machine's Settings window and select the new interface in the "Interface name" drop-down list. With the above example, this drop-down list would contain "VM1 external".

If your host is running **Windows XP or newer**, you can also use the built-in bridging feature to connect your host interfaces to your physical network card. After creating the desired host interfaces, select your physical network adapter in the Network Connections folder and the desired host interface adapters and select "Bridge connections" from the popup menu. Note that you have to transfer your network configuration from your physical network adapter to the network bridge as mentioned above, because your physical network adapter will only function as a transport medium in your bridge setup. When more than one connection is active on a bridge, Windows will automatically put your physical Ethernet adapter into promiscuous mode so that it will receive network data for all bridged connections.

6.7 Host Interface Networking and bridging on Linux hosts

Before you proceed, please read chapter 6.5, *Introduction to Host Interface Networking (HIF)*, page 73.

> **Note:** There were some changes to the way dynamic host interface configuration is done in VirtualBox 1.4.0, due to changes in Linux kernel versions 2.6.18 and later. Also, this entire section of the manual was rewritten for Virtual 1.4.0. Please reread these sections if you used dynamic interfaces on earlier versions.

Since the Linux kernel has built-in support for virtual network devices (so-called TAP interfaces), VirtualBox on Linux makes use of these instead of providing custom

host networking drivers. The TAP interfaces behave like physical network interfaces on your host and will work with any networking tools installed on your host system. From the point of view of the host, it looks like the guest's network card is connected to the TAP interface with a network cable. In order to use Host Interface Networking in VirtualBox, you must have access to the device `/dev/net/tun`. Check which group this device belongs to and make sure that any users who need access to VirtualBox Host Networking are members of this group. In most cases, this device will belong to the `vboxusers` group.

On Linux hosts, you have a choice of creating *permanent* networking interfaces which guests can attach to when they are created or having VirtualBox create a *dynamic* interface for a guest when the guest is started and remove it when the guest is stopped. Permanent interfaces are more suitable for hosts with a known set of guests that does not change often (such as some server setups), and they are easier to set up. Having VirtualBox create the interfaces dynamically provides more flexibility, but will normally require you to enter an administrator password each time an interface is created or removed.

6.7.1 Permanent host interfaces and bridging

On Linux hosts, setting up a permanent host interface using bridging typically consists of three steps:

1. First, you must create a bridge on the host and add one of the host's physical network interfaces to it, usually `eth0`. This will let you connect that interface to the virtual interfaces used by the virtual machines.

   Keep in mind that bridging is an Ethernet concept, not a TCP/IP one. In physical networking, bridging is normally used to connect two Ethernet networks, letting computers on the one communicate with computers on the other through a single point of contact without having to merge the networks into one.[1]

2. For each guest network card that uses host interface networking, you must create a new "virtual" host interface (usually called `vbox0` or similar) and add this interface to the bridge.

3. Finally, specify the name of the new host interface in the settings of the virtual machine's virtual network card.

Unfortunately, Linux distributions differ substantially in how networking is configured. As we cannot provide instructions for all Linux distributions, we have restricted ourselves to describing how to set up bridging on Debian, Ubuntu, Fedora/Red Hat and openSUSE; in addition, we offer some generic instructions for advanced users.

[1] A useful introduction to bridging can be found here: http://gentoo-wiki.com/HOWTO_setup_a_gentoo_bridge. While this is targeted at a Gentoo system, it contains a useful generic introduction.

VirtualBox ships with two utilities, `VBoxAddIF` and `VBoxDeleteIF`, which work on all distributions. These tools allow you to create and delete permanent host interfaces (which will automatically be recreated every time you boot your host computer) and optionally add them to an existing bridge. We also provide a utility called `VBoxTunctl` which you can use to create a temporary interface. These tools are described in chapter 6.7.1.5, *Host Interface Networking utilities for Linux*, page 81. Even if you use plan to use host interfaces to create other networking setups than what we describe here, we recommend that you read the following instructions in order to get an understanding of how the interfaces work.

Some distributions – such as Debian and Ubuntu – have built-in tools to create host interfaces; you may also use those tools on those distributions.

6.7.1.1 Debian and Ubuntu hosts

To set up a permanent host interface on a modern Debian or Ubuntu host, follow these steps:

1. First install the bridge utilities (`bridge-utils`). package. You can do this from the command line as follows:

   ```
   sudo apt-get install bridge-utils
   ```

2. Next, you must add an entry to the file `/etc/network/interfaces` to describe the bridge. The following sample entry creates a bridge called `br0`, adds the host ethernet interface `eth0` to it and tells it to obtain an IP address using DHCP so that the host remains able to access the network.

   ```
   auto br0
   iface br0 inet dhcp
       bridge_ports eth0
   ```

   You will probably want to change this to suit your own networking needs. In particular, you may want to assign a static IP address to the bridge. You will find more documentation in the files

   a) `/usr/share/doc/bridge-utilities/README.Debian.gz` and

   b) `/usr/share/doc/ifupdown/examples/network-interfaces.gz`.

3. Restart networking on the host:

   ```
   sudo /etc/init.d/networking restart
   ```

   After this the bridge will be recreated every time you boot your host system.

4. Now, to create a permanent host interface called `vbox0` (all host interfaces created in this way must be called `vbox` followed by a number) and add it to the network bridge created above, use the following command (see chapter 6.7.1.5, *Host Interface Networking utilities for Linux*, page 81 for more details):

```
sudo VBoxAddIF vbox0 <user> br0
```

Replace `<user>` with the name of the user who is supposed to be able to use the new interface.

To tell VirtualBox to use the interface, select the virtual machine which is to use it in the main window of the VirtualBox application, configure one of its network adapters to use Host Interface Networking (using "Settings", "Network", "Attached to") and enter `vbox0` into the "Interface name" field. You can only use a given interface (`vbox0`, `vbox1` and so on) with a single virtual network adapter.

Alternatively, you can use the VBoxManage command line tool (in this example we are attaching the interface to the first network card of the virtual machine "My VM"):

```
VBoxManage modifyvm "My VM" -hostifdev1 vbox0
```

To set up a host interface using Debian and Ubuntu's native methods, do the following instead of step 4 above:

1. First install the User Mode Linux utilities package (`uml-utilities`), which contains tools to create TAP interfaces. You can do this from the command line as follows:

   ```
   sudo apt-get install uml-utilities
   ```

   In order for VirtualBox to be able to access the interface, the user who will be running the virtual machine must be added to the group `uml-net`, for example with the following command (replace `<user>` with your user name):

   ```
   sudo gpasswd -a <user> uml-net
   ```

   You will have to log in again for the change to take effect.

2. To describe the TAP interface to your Debian or Ubuntu system, add an entry to the file `/etc/network/interfaces`. This names the the interface and must also specify the user who will be running the virtual machine using the interface.

   The following sample entry creates the interface `tap0` for the user `<user>` (again, replace with your user name):

   ```
   auto tap0
   iface tap0 inet manual
       up ifconfig $IFACE 0.0.0.0 up
       down ifconfig $IFACE down
       tunctl_user <user>
   ```

   You will probably want to change the entry based on your networking needs. The file `/usr/share/doc/uml-utilities/README.Debian` on your host computer will have additional documentation.

3. To add the TAP interface to the bridge, replace the line

   ```
   bridge_ports eth0
   ```

   in the bridge section in `/etc/network/interfaces` with

   ```
   bridge_ports eth0 tap0
   ```

4. Restart networking on the host:

   ```
   sudo /etc/init.d/networking restart
   ```

6.7.1.2 Bridging on openSUSE hosts

The following instructions explain how to create bridge on openSUSE. Note that bridging on openSUSE hosts may not work properly if you are using NetworkManager to manage your network connections. To create a bridge on a recent openSUSE host, you must first install the bridge utilities (`bridge-utils`) package. If you are working from the command line this can be done as follows:

```
sudo /sbin/yast -i bridge-utils
```

Then you must create a text file describing the bridge to be created. The name of the file must correspond to the name of the bridge you wish to create. To create the bridge `br0`, you should call the file `/etc/sysconfig/network/ifcfg-br0`. Below we have given an example of a file which creates a bridge including the network device `eth0`, obtains an IP address by DHCP (through the network device) and is started automatically when openSUSE starts. You will probably want to adjust this to match your networking requirements.

```
BOOTPROTO='dhcp'
NETMASK='255.255.255.0'
STARTMODE='auto'
USERCONTROL='no'
DHCLIENT_TIMEOUT=30
BRIDGE='yes'
BRIDGE_PORTS='eth0'
```

For this example to work, you will also need to change the configuration for the network interface `eth0` to a static IP address of 0.0.0.0, as openSUSE does not do this automatically when the interface is added to the bridge. You can do this using the graphical interface or by changing the following settings in the file `/etc/sysconfig/network/ifcfg-eth-xx:xx:xx:xx:xx:xx`, where the last part should be replaced with the hardware address of the network card.

```
BOOTPROTO='static'
IPADDR='0.0.0.0'
```

You can activate the bridge immediately after creating it with the command:

```
sudo /sbin/ifdown eth0
sudo /sbin/ifup br0
```

The bridge will be activated automatically from now on when the host is restarted.

Now, to create a permanent host interface called `vbox0` (all host interfaces created in this way must be called `vbox` followed by a number) and add it to the network bridge created above, use the following command (see chapter 6.7.1.5, *Host Interface Networking utilities for Linux*, page 81 for more details):

```
sudo VBoxAddIF vbox0 <user> br0
```

Replace `<user>` with the name of the user who is supposed to be able to use the new interface.

To tell VirtualBox to use this interface (`vbox0`) for a virtual machine, select the VM in the main window, configure one of its network adaptors to use Host Interface Networking (using “Settings”, “Network”, “Attached to”) and enter “vbox0” into the “Interface name” field. You can only use a given interface (`vbox0`, `vbox1` and so on) with a single virtual machine.

Alternatively, you can use the VBoxManage command line tool (in this example we are attaching the interface to the first network card of the virtual machine “My VM”:

```
VBoxManage modifyvm "My VM" -hostifdev1 vbox0
```

6.7.1.3 Bridging on Redhat and Fedora hosts

To create a bridge on Redhat and Fedora, you must first install the bridge utilities (`bridge-utils`) package. Then you must create a configuration file describing the bridge you wish to create. The following is the contents of an example configuration file `/etc/sysconfig/network-scripts/ifcfg-br0`, which sets the bridge br0 to get its IP address using DHCP and to start automatically when the system is started. You will probably want to adjust this to match your networking requirements.

```
DEVICE=br0
TYPE=Bridge
BOOTPROTO=dhcp
ONBOOT=yes
```

To add the network card `eth0` to the bridge, add the following line to the end of the file `/etc/sysconfig/network-scripts/ifcfg-eth0`:

```
BRIDGE=br0
```

You can activate the bridge immediately after creating it with the command:

```
sudo /sbin/service network restart
```

The bridge will be activated automatically from now on when the host is restarted.

Now, to create a permanent host interface called `vbox0` (all host interfaces created in this way must be called `vbox` followed by a number) and add it to the network bridge created above, use the following command (see chapter 6.7.1.5, *Host Interface Networking utilities for Linux*, page 81 for more details):

```
sudo VBoxAddIF vbox0 <user> br0
```

Replace `<user>` with the name of the user who is supposed to be able to use the new interface.

To tell VirtualBox to use this interface (`vbox0`) for a virtual machine, select the VM in the main window, configure one of its network adaptors to use Host Interface Networking (using "Settings", "Network", "Attached to") and enter "vbox0" into the "Interface name" field. You can only use a given interface (`vbox0`, `vbox1` and so on) with a single virtual machine.

Alternatively, you can use the VBoxManage command line tool (in this example we are attaching the interface to the first network card of the virtual machine "My VM":

```
VBoxManage modifyvm "My VM" -hostifdev1 vbox0
```

6.7.1.4 Bridging with other distributions

Most modern Linux distributions provide their own way of setting up ethernet bridges. We recommend that you follow the instructions provided by your distribution to do this. For distributions which do not provide their own method of creating bridges, we provide generic instructions below. Please ensure that you thoroughly understand how your distribution's networking scripts work before following these instructions, as they involve making changes to your host network configuration in ways normally only done by the networking scripts, and as such may interfere with your network setup.

First of all, you will need to install the bridge utilities (usually named `bridge-utils` or similar). Once installed, as `root`, follow these instructions to create and configure a bridge:

1. Create a new bridge with this command:

   ```
   brctl addbr br0
   ```

2. If you are not using DHCP, run `ifconfig` and note down the network configuration of your existing network interface (e.g. `eth0`), which we will need to copy to the bridge in a minute.

3. Remove the IP address configuration from the existing network device (e.g. `eth0`) with:

   ```
   ifconfig eth0 0.0.0.0
   ```

   Warning: You will lose network connectivity on `eth0` at this point.

4. Add your network adapter to the bridge:

   ```
   brctl addif br0 eth0
   ```

5. Transfer the network configuration previously used with your physical ethernet adapter to the new bridge. If you are using DHCP, this should work:

```
dhclient br0
```

Otherwise, run `ifconfig br0 x.x.x.x netmask x.x.x.x` and use the values that you noted down previously.

6. To create a permanent host interface called `vbox0` (all host interfaces created in this way must be called `vbox` followed by a number) and add it to the network bridge created above, use the following command (see chapter 6.7.1.5, *Host Interface Networking utilities for Linux*, page 81 for more details):

```
VBoxAddIF vbox0 <user> br0
```

Replace `<user>` with the name of the user who is supposed to be able to use the new interface.

6.7.1.5 Host Interface Networking utilities for Linux

Although Linux comes with built-in support for virtual networking interfaces, there are not many programs available for managing these. VirtualBox supplies three tools for this purpose: `VBoxAddIF`, `VBoxDeleteIF` and `VBoxTunctl`. The last of these is in fact the `tunctl` utility from the User Mode Linux project. In this section, we describe how to use these utilities.

`VBoxAddIF` creates a permanent TAP interface which does not go away when the host system is restarted. This interface should be called `vbox0`, `vbox1` or similar. The following command creates the interface `vbox0` for the user `<user>` and adds it to the bridge `br0`. If you do not wish to add the interface to a bridge, you can leave off the bridge name.

```
sudo VBoxAddIF vbox0 <user> br0
```

Change the interface, user and bridge names to fit your own setup.

To remove an interface which you have created with `VBoxAddIF`, use the following command. Replace `vbox0` with the name of the interface.

```
sudo VBoxDeleteIF vbox0
```

To create a temporary TAP interface which will disappear when the host system is restarted, use the `VBoxTunctl` command. The following example creates the interface `vbox0` for the user `<user>`:

```
sudo VBoxTunctl -t vbox0 -u <user>
```

If you have installed the bridge utilities (see the preceding sections), you can add this temporary interface to an Ethernet bridge using the command

```
sudo brctl addif br0 vbox0
```

Replace `br0` with the name of the bridge and `vbox0` with the name of the interface. Before you can use the interface, you will still need to make it active (or "bring it up" in networking terminology, usually using the standard Linux `ifconfig` utility) and configure it with an IP address and related information. To remove a temporary interface, do the following, replacing `vbox0` with the name of the interface to be removed:

```
sudo VBoxTunctl -d vbox0
```

6.7.2 Creating interfaces dynamically when a virtual machine starts up

As an alternative to the permanent interfaces described previously, you can tell VirtualBox to execute commands (usually scripts) to set up your network dynamically, every time a virtual machine starts or stops. This is normally done in order to create the TAP interfaces at VM startup time, although you can also use this feature to configure existing interfaces. If you are not using permanent interfaces then the startup command should write the name of the interface which it has created, typically something like `tap0` or `tap2`, to its standard output (the `VBoxTunctl -b` command does exactly this) and the command executed when the machine stops should remove the interface again.

The commands and scripts used will depend on the networking configuration that you want to set up. Both commands are given a file descriptor to the Linux TAP device as their first argument (this is only valid if the virtual machine is using previously created interfaces) and the name of the interface, if it is known, as the second argument. In most circumstances, you will only want to use the second argument.

Here is an example of a set up script which creates a TAP interface and adds it to the network bridge `br0`.

```
#!/bin/bash

# Create an new TAP interface for the user 'vbox' and remember its name.
interface=`VBoxTunctl -b -u vbox`

# If for some reason the interface could not be created, return 1 to
# tell this to VirtualBox.
if [ -z "$interface" ]; then
exit 1
fi

# Write the name of the interface to the standard output.
echo $interface

# Bring up the interface.
/sbin/ifconfig $interface up

# And add it to the bridge.
/sbin/brctl addif br0 $interface
```

If this script is saved as `/home/vbox/setuptap.sh` and made executable, it can be used to create a TAP interface when a virtual machine is started, by configuring one of the machines network adapters to use Host Interface Networking (without specifying a device in the "Interface Name" field) and entering `gksudo /home/vbox/setuptap.sh` into the "Setup Application" field (replace `gksudo` by `kdesu`, or whatever is appropriate for your system). Alternatively you can use the the VBoxManage command line tool (in the following example for a machine called "Linux VM"):

```
VBoxManage modifyvm "Linux VM" -tapsetup1 "gksudo /home/vbox/setuptap.sh"
```

An example of a matching script to remove the interface from the bridge and shut it down would be:

```
#!/bin/bash

# Remove the interface from the bridge.  The second script parameter is
# the interface name.
/sbin/brctl delif br0 $2

# And use VBoxTunctl to remove the interface.
VBoxTunctl -d $2
```

If this is saved as `/home/vbox/cleanuptap.sh` and made executable, the virtual machine can be told to execute it when it shuts down by entering `gksudo /home/vbox/cleanuptap.sh`, into the "Termination Application" field in VirtualBox's network configuration settings, or by using VBoxManage:

```
VBoxManage modifyvm "Linux VM" -tapterminate1
"gksudo /home/vbox/cleanuptap.sh"
```

Note: The `VBoxSDL` front end to VirtualBox (see chapter 7.3, *VBoxSDL, the simplified VM displayer*, page 89) allows for an additional way of configuring TAP interfaces if it is started from a custom parent process. This parent process can allocate the required TAP interfaces and let VirtualBox inherit the file handles. For this to work, the file descriptor has to be passed to `VBoxSDL` using the option `-tapfd<N> <fd>`. In this case, the setup and termination scripts will not be called.

6.8 Host Interface Networking and bridging on OpenSolaris hosts

Before you proceed, please read chapter 6.5, *Introduction to Host Interface Networking (HIF)*, page 73.

VirtualBox uses OpenSolaris' Crossbow architecture for Host Interface Networking. Currently, Crossbow does not support permanent virtual network interfaces. Thus, all virtual network interfaces created will be lost when the host reboots.

Note: Creating and using virtual network interfaces requires root permissions. This issue with Crossbow is likely to be resolved in subsequent OpenSolaris releases.

6.8.1 Creating interfaces using the setup script

The easiest way to create a virtual network interface is by using the supplied `vnic_setup.sh` script. You need to supply the script with a valid MAC address to assign to the virtual network interface. For example:

```
/opt/VirtualBox/vnic_setup.sh 0:1:4a:f2:31:34
```

Note that each virtual network interface must be assigned a different MAC address. The script creates a new virtual network interface, assigns its MAC address, and assigns a destination IP address which will be used by the guest. The newly created interface will be bridged to the first active physical ethernet interface on your host.

Crossbow assigns the names of virtual network interfaces as "vnic" followed by the ID, such as "vnic900".

After executing the setup script you could use `ifconfig -a` to verify the virtual network interface is up and running. You should get an output that resembles this:

```
vnic900: flags=201000851 UP,POINTOPOINT,RUNNING,MULTICAST,IPv4,CoS mtu 1500 index 4
inet 192.168.1.100 --> 192.168.1.200 netmask ffffff00
ether 0:1:4a:f2:31:34
```

If you are using the VirtualBox graphical interface, you can use this script as the "Setup Application" and use the `vnic_term.sh` as the "Terminate Application" in the "Network" configuration of your virtual machine. Be sure to leave the "TAP interface" name field as blank. By leaving the name blank you instruct the script to create a new interface which VirtualBox would use. If you specify a name, the script assumes the interface already exists and could be modified to configure an existing interface.

Once the virtual network interface is up and running as described above, VirtualBox can now use it for transferring data using the virtual network card as if it were a physical one. For all practical purposes, the virtual network card now behaves like a physical one.

6.8.2 Creating interfaces manually

Although the script does most of the work of setting up a ready-to-use virtual network interface, you might want to do so yourself for maximum flexibility.

Using the Crossbow tool `vna` it is possible to create and delete virtual network interfaces. To create one you need to supply the physical ethernet interface and the MAC address. For example:

```
/usr/lib/vna iprb0 00:01:4a:f2:31:34
```

In the above example, "iprb0" refers to the link name of the physical ethernet interface on the host that bridges to the virtual network interface. To get a list of link names for network devices on your host, use `dladm show-link`.

If `vna` successfully created the virtual network interface, it prints the name of the newly created interface. Next you need to 'plumb' the virtual interface. For this execute `ifconfig vnic900 plumb` and replace "vnic900" with the name of the new interface. Now the interface is up and running. Additionally, to use the interface from the guest you would need to configure it using using traditional `ifconfig` methods. For more information, refer the `ifconfig` man page and `vnic_setup.sh`.

The created interface can now be used with VirtualBox by specifying the interface name in the virtual machine configuration. If necessary, you could do last minute configuration of your virtual network interface by using a custom script and specify it as the "Setup Application". VirtualBox would invoke the setup application and pass it the MAC address and the virtual network interface name as parameters.

To delete a virtual network interface, first you must unplumb it if it has been plumbed before, and then execute `/usr/lib/vna` followed by the ID of the interface. This will remove a virtual network interface and relinquish its ID.

6.9 Internal networking

Internal Networking is similar to host interface networking in that the VM can directly communicate with the outside world. However, the "outside world" is limited to other VMs which connect to the same internal network.

Even though technically, everything that can be done using internal networking can also be done using host interface networking, there are two good reasons why this extra mode was implemented:

1. **Security.** In host interface networking mode, all traffic goes through an interface of the host system. It is therefore possible to attach a packet sniffer (such as Ethereal) to the host interface and log all traffic that goes over a given interface. 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, Host Interface Networking therefore is not an option.

2. **Speed.** Internal networking is more efficient than host interface networking, as VirtualBox can directly transmit the data without having to send it through the host operating system's networking stack.

Internal networks are created automatically as needed, i.e. 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 VirtualBox support driver will automatically "wire" the cards and act as a network switch. The 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:

- You can use a VM's "Settings" dialog in the VirtualBox graphical user interface. In the "Networking" category of the settings dialog, select "Internal Networking" from the drop-down list of networking modes. Now select the name of an existing internal network from the drop-down below or enter a new name into the entry field.

- You can use `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. See also chapter 8.5, *VBoxManage modifyvm*, page 102.

In any case, you will have to configure the (virtual) network cards in the guest operating systems that are participating in the internal network to use static IP addresses (because the internal network provided by VirtualBox does not support DHCP, like VirtualBox's NAT engine would). These IP addresses should use IP addresses on the same subnet (e.g. 192.168.2.1 and 192.168.2.2). You may have to deactivate guest firewalls in order to allow guests to communicate with each other.

As a security measure, the Linux implementation of internal networking only allows VMs running under the same user ID to establish an internal network.

7 Alternative front-ends; remote virtual machines

7.1 Introduction

As briefly mentioned in chapter 1.2, *Features overview*, page 10, VirtualBox has a very flexible internal design that allows you to use different front-ends to control the same virtual machines. To illustrate, you can, for example, start a virtual machine with VirtualBox's easy-to-use graphical user interface and then stop it from the command line. With VirtualBox's support for the Remote Desktop Protocol (VRDP), you can even run virtual machines remotely on a headless server and have all the graphical output redirected over the network.

In detail, the following front-ends are shipped in the standard VirtualBox package:

1. `VirtualBox` is our graphical user interface (GUI), which most of this User Manual is dedicated to describing, especially in chapter 3, *Starting out with VirtualBox*, page 27. While this is the easiest-to-use of our interfaces, it does not (yet) cover all the features that VirtualBox provides. Still, this is the best way to get to know VirtualBox initially.

2. `VBoxManage` is our command-line interface and is described in the next section.

3. `VBoxSDL` is an alternative, simple graphical front-end with an intentionally limited feature set, designed to only display virtual machines that are controlled in detail with `VBoxManage`. This is interesting for business environments where displaying all the bells and whistles of the full GUI is not feasible. `VBoxSDL` is described in chapter 7.3, *VBoxSDL, the simplified VM displayer*, page 89.

4. Finally, `VBoxHeadless` is yet another front-end that produces no visible output on the host at all, but merely acts as a VRDP server. Now, even though the other graphical front-ends (VirtualBox and VBoxSDL) also have VRDP support built-in and can act as a VRDP server, this particular 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. For details, see chapter 7.4.1, *VBoxHeadless, the VRDP-only server*, page 91.

If the above front-ends still do not satisfy your particular needs, it is relatively painless to create yet another front-end to the complex virtualization engine that is the core of VirtualBox, as the VirtualBox core neatly exposes all of its features in a clean COM/XPCOM API. This API is described in chapter 10, *VirtualBox programming interfaces*, page 127.

7.2 Using VBoxManage to control virtual machines

This section will give you a brief introduction to VBoxManage and how you can use it to create and operate virtual machines.

In essence, VBoxManage supports everything that our graphical user interface allows you to do with the click of a button. VBoxManage supports a lot more than that, however. It exposes really all the features of the virtualization engine, even those that cannot (yet) be accessed from the GUI.

You will need to use the command line if you want to

- use a different user interface than the main GUI (for example, VBoxSDL or 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 vms" or "createvm" or "startvm". All the subcommands that `VBoxManage` supports are described in detail in chapter 8, *VBoxManage reference*, page 96.

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 VirtualBox GUI. Note that if that name contains spaces, then you must enclose the entire name in double quotes (as it 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 VirtualBox uses to refer to the virtual machine. Assuming that the aforementioned VM called "Windows XP" has the UUID shown below, the following command has the same effect as the previous:

  ```
  VBoxManage startvm 670e746d-abea-4ba6-ad02-2a3b043810a5
  ```

You can type `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 VirtualBox from the command line are listed below:

- To create a new virtual machine from the command line and immediately register it with VirtualBox, use `VBoxManage createvm` with the `-register` option,[1] like this:

[1] For details, see chapter 8.4, *VBoxManage createvm*, page 101.

```
$ VBoxManage createvm -name "SUSE 10.2" -register
VirtualBox Command Line Management Interface Version 1.6.2
(C) 2005-2008 Sun Microsystems, Inc.
All rights reserved.

Virtual machine 'SUSE 10.2' is created.
UUID: c89fc351-8ec6-4f02-a048-57f4d25288e5
Settings file: '/home/username/.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.

- To show the configuration of a particular VM, use `VBoxManage showvminfo`; see chapter 8.2, *VBoxManage showvminfo*, page 100 for details and an example.

- To change VM settings, use `VBoxManage modifyvm`, e.g. as follows:

```
VBoxManage modifyvm "Windows XP" -memory "512MB"
```

For details, see chapter 8.5, *VBoxManage modifyvm*, page 102.

- To control VM operation, use one of the following:

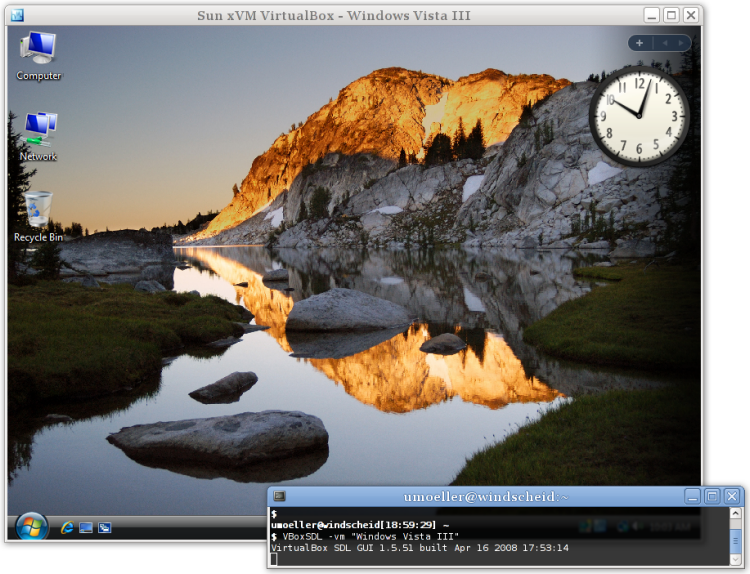
  - To start a VM that is currently powered off, use `VBoxManage startvm`; see chapter 8.6, *VBoxManage startvm*, page 107 for details.

  - To pause or save a VM that is currently running, use `VBoxManage controlvm`; see chapter 8.7, *VBoxManage controlvm*, page 107 for details.

7.3 VBoxSDL, the simplified VM displayer

VBoxSDL is a simple graphical user interface (GUI) that lacks the nice point-and-click support which VirtualBox, our main GUI, provides. VBoxSDL is currently primarily used internally for debugging VirtualBox and therefore not officially supported. Still, you may find it useful for environments where the virtual machines are not necessarily controlled by the same person that uses the virtual machine.

As you can see in the following screenshot, VBoxSDL does indeed only provide a simple window that contains only the “pure” virtual machine, without menus or other controls to click upon and no additional indicators of virtual machine activity:



To start a virtual machine with VBoxSDL instead of the VirtualBox GUI, enter the following on a command line:

```
VBoxSDL -vm <vm>
```

where `<vm>` is, as usual with VirtualBox command line parameters, the name or UUID of an existing virtual machine.

7.4 Remote virtual machines (VRDP support)

VirtualBox, the graphical user interface, has a built-in server for the VirtualBox Remote Desktop Protocol (VRDP). This allows you to see the output of a virtual machine's window remotely on any other computer and control the virtual machine from there, as if it was running on the remote machine.

VRDP is a backwards-compatible extension to Microsoft's Remote Desktop Protocol (RDP). Typically graphics updates and audio are sent from the remote machine to the client, while keyboard and mouse events are sent back.

With `VirtualBox`, the graphical user interface, the VRDP server is disabled by default, but can easily be enabled on a per-VM basis either with the VirtualBox GUI or with `VBoxManage`:

```
VBoxManage modifyvm <vmname> -vrdp on
```

If you use `VBoxHeadless` (described below), VRDP support will be automatically enabled.

Additional settings for `modifyvm` are `-vrdpport` and `-vrdpauthtype`; see chapter 8.5, *VBoxManage modifyvm*, page 102 for details.

7.4.1 VBoxHeadless, the VRDP-only server

While the VRDP server that is built into the `VirtualBox` GUI is perfectly capable of running virtual machines remotely, it is not convenient to have to run `VirtualBox` if you never want to have VMs displayed locally in the first place. In particular, if you are running servers whose only purpose is to host VMs, and all your VMs are supposed to run remotely over VRDP, then it is pointless to have a graphical user interface on the server at all – especially since, on a Linux or Solaris host, `VirtualBox` comes with dependencies on the Qt and SDL libraries, which is inconvenient if you would rather not have the X Window system on your server at all.

VirtualBox therefore comes with yet another front-end that produces no visible output on the host at all, but instead only delivers VRDP data. With VirtualBox 1.6, this "headless server" is now aptly called `VBoxHeadless`. (In previous versions, it was called `VBoxVRDP`. For the sake of backwards compatibility, the VirtualBox installation still installs an executable with that name as well.)

To start a virtual machine with VBoxHeadless, you have two options:

- You can use `VBoxManage startvm <vmname> -type vrdp`. The extra `-type` option causes the VirtualBox core to use `VBoxHeadless` as the front-end to the internal virtualization engine.

- The recommended way, however, is to use `VBoxHeadless` directly, as follows:

  ```
  VBoxHeadless -startvm <uuid|name>
  ```

  This is the recommended way, because when starting the headless interface through `VBoxManage`, you will not be able to view or log messages that `VBoxHeadless` may have output on the console. Especially in case of startup errors, such output might be desirable for problem diagnosis.

7.4.2 Step by step: creating a virtual machine on a headless server

The following instructions may give you an idea how to create a virtual machine on a headless server over a network connection. We will create a virtual machine, establish a VRDP connection and install a guest operating system – all without having to touch the headless server. All you need is the following:

1. VirtualBox on a server machine with a supported host operating system; for the following example, we will assume a Linux server;

2. an ISO file on the server, containing the installation data for the guest operating system to install (we will assume Windows XP in the following example);

3. a terminal connection to that host over which you can access a command line (e.g. via `telnet` or `ssh`);

4. an RDP viewer on the remote client; on a Linux client, you could use `rdesktop` to connect; from a Windows machine, you could use the RDP viewer that comes with Windows (usually found in “Accessories” -> “Communication” -> “Remote Desktop Connection”).

Note again that on the server machine, since we will only use the headless server, neither Qt nor SDL nor the X Window system will be needed.

1. On the headless server, create a new virtual machine:

   ```
   VBoxManage createvm -name "Windows XP" -register
   ```

   Note that if you do not specify `-register`, you will have to manually use the registervm command later.

2. Make sure the settings for this VM are appropriate for the guest operating system that we will install. For example:

   ```
   VBoxManage modifyvm "Windows XP" -memory "256MB" \
            -acpi on -boot1 dvd -nic1 nat
   ```

3. Create a virtual hard disk for the VM (in this case, 10GB in size) and register it with VirtualBox:

   ```
   VBoxManage createvdi -filename "WinXP.vdi" -size 10000 -register
   ```

4. Set this newly created VDI file as the first virtual hard disk of the new VM:

   ```
   VBoxManage modifyvm "Windows XP" -hda "WinXP.vdi"
   ```

5. Register the ISO file that contains the operating system installation that you want to install later:

   ```
   VBoxManage registerimage dvd /full/path/to/iso.iso
   ```

6. Attach this ISO to the virtual machine, so it can boot from it:

   ```
   VBoxManage modifyvm "Windows XP" -dvd /full/path/to/iso.iso
   ```

   (Alternatively, you can use `VBoxManage controlvm dvdattach` directly, without having to register the image first; see chapter 8.7, *VBoxManage controlvm*, page 107 for details.)

7. Start the virtual machine using VBoxHeadless:

   ```
   VBoxHeadless -startvm "Windows XP"
   ```

   If everything worked, you should see a copyright notice. If, instead, you are returned to the command line, then something went wrong.

8. On the client machine, fire up the RDP viewer and try to connect to the server. Assuming a Linux client, try the following:

   ```
   rdesktop -a 16 my.host.address
   ```

   (With rdesktop, the -a 16 option requests a color depth of 16 bits per pixel, which we recommend. Also, after installation, you should set the color depth of your guest operating system to the same value.)

   You should now be seeing the installation routine of your guest operating system.

7.4.3 Remote USB

As a special feature on top of the VRDP support, VirtualBox supports remote USB devices over the wire as well. That is, the VirtualBox guest that runs on one computer can access the USB devices of the remote computer on which the RDP data is being displayed the same way as USB devices that are connected to the actual host. This allows for running virtual machines on a VirtualBox host that acts as a server, where a client can connect from elsewhere that needs only a network adapter and a display capable of running an RDP viewer. When USB devices are plugged into the client, the remote VirtualBox server can access them.

For these remote USB devices, the same filter rules apply as for other USB devices, as described with chapter 3.7.6.1, *USB settings*, page 48. All you have to do is specify "Remote" (or "Any") when setting up these rules.

7.4.4 RDP authentication

For each virtual machine that is remotely accessible via RDP, you can individually determine if and how RDP connections are authenticated.

For this, use `VBoxManage modifyvm` command with the `-vrdpauthtype` option; see chapter 8.5, *VBoxManage modifyvm*, page 102 for a general introduction. Three methods of authentication are available:

- The "null" method means that there is no authentication at all; any client can connect to the VRDP server and thus the virtual machine. This is, of course, very insecure and only to be recommended for private networks.

- The "external" method provides external authentication through a special authentication library.

  VirtualBox comes with two default libraries for external authentication:

  - On Linux hosts, `VRDPAuth.so` authenticates users against the host's PAM system.
  - On Windows hosts, `VRDPAuth.dll` authenticates users against the host's WinLogon system.

In other words, the "external" method per default performs authentication with the user accounts that exist on the host system.

However, you can replace the default "external" authentication module with any other module. For this, VirtualBox provides a well-defined interface that allows you to write your own authentication module; see chapter 9.3, *Custom external VRDP authentication*, page 116 for details.

- Finally, the "guest" authentication method performs authentication with a special component that comes with the Guest Additions; as a result, authentication is not performed with the host users, but with the guest user accounts. This method is currently still in testing and not yet supported.

7.4.5 RDP encryption

RDP features data stream encryption, which is based on the RC4 symmetric cipher (with keys up to 128bit). The RC4 keys are being replaced in regular intervals (every 4096 packets).

RDP provides three different authentication methods:

1. Historically, RDP4 authentication was used, with which the RDP client does not perform any checks in order to verify the identity of the server it connects to. Since user credentials can be obtained using a man in the middle (MITM) attack, RDP4 authentication is insecure and should generally not be used.

2. RDP5.1 authentication employs a server certificate for which the client possesses the public key. This way it is guaranteed that the server possess the corresponding private key. However, as this hard-coded private key became public some years ago, RDP5.1 authentication is also insecure and cannot be recommended.

3. RDP5.2 authentication is based on TLS 1.0 with customer-supplied certificates. The server supplies a certificate to the client which must be signed by a certificate authority (CA) that the client trusts (for the Microsoft RDP Client 5.2, the CA has to be added to the Windows Trusted Root Certificate Authorities database). VirtualBox allows you to supply your own CA and server certificate and uses OpenSSL for encryption.

While VirtualBox supports all of the above, only RDP5.2 authentication should be used in environments where security is a concern. As the client that connects to the server determines what type of encryption will be used, with rdesktop, the Linux RDP viewer, use the `-4` or `-5` options.

7.4.6 VRDP multiple connections

The VirtualBox built-in RDP server supports simultaneous connections to the same running VM from different clients. All connected clients see the same screen output

and share a mouse pointer and keyboard focus. This is similar to several people using the same computer at the same time, taking turns at the keyboard.

The following command enables multiple connection mode:

```
VBoxManage modifyvm VMNAME -vrdpmulticon on
```

If the guest uses multiple monitors then multiple connection mode must be active in order to use them at the same time (see chapter 9.6, *Multiple monitors for the guest*, page 119).

8 VBoxManage reference

When running VBoxManage without parameters or when supplying an invalid command line, the below syntax diagram will be shown. Note that the output will be slightly different depending on the host platform; when in doubt, check the output of `VBoxManage` for the commands available on your particular host.

```
Usage:

VBoxManage [-v|-version]    print version number and exit
VBoxManage -nologo ...      suppress the logo

VBoxManage -convertSettings ...        allow to auto-convert settings files
VBoxManage -convertSettingsBackup ...  allow to auto-convert settings files
                                       but create backup copies before
VBoxManage -convertSettingsIgnore ...  allow to auto-convert settings files
                                       but don't explicitly save the results

VBoxManage list             vms|runningvms|ostypes|hostdvds|hostfloppies|
hostifs|                            hdds|dvds|floppies|usbhost|usbfilters|
                            systemproperties

VBoxManage showvminfo       <uuid>|<name>
                            [-details]
                            [-statistics]
                            [-machinereadable]

VBoxManage registervm       <filename>

VBoxManage unregistervm     <uuid>|<name>
                            [-delete]

VBoxManage createvm         -name <name>
                            [-register]
                            [-basefolder <path> | -settingsfile <path>]
                            [-uuid <uuid>]


VBoxManage modifyvm         <uuid|name>
                            [-name <name>]
                            [-ostype <ostype>]
                            [-memory <memorysize>]
                            [-vram <vramsize>]
                            [-acpi on|off]
                            [-ioapic on|off]
                            [-pae on|off]
                            [-hwvirtex on|off|default]
                            [-monitorcount <number>]
```

```
                        [-bioslogofadein on|off]
                        [-bioslogofadeout on|off]
                        [-bioslogodisplaytime <msec>]
                        [-bioslogoimagepath <imagepath>]
                        [-biosbootmenu disabled|menuonly|messageandmenu]
                        [-biossystemtimeoffset <msec>]
                        [-biospxedebug on|off]
                        [-boot<1-4> none|floppy|dvd|disk|net>]
                        [-hd<a|b|d> none|<uuid>|<filename>]
                        [-sata on|off]
                        [-sataportcount <1-30>]
                        [-sataport<1-30> none|<uuid>|<filename>]
                        [-sataideemulation<1-4> <1-30>]
                        [-dvd none|<uuid>|<filename>|host:<drive>]
                        [-dvdpassthrough on|off]
                        [-floppy disabled|empty|<uuid>|
                                 <filename>|host:<drive>]
                        [-nic<1-N> none|null|nat|hostif|intnet]
                        [-nictype<1-N> Am79C970A|Am79C973|82540EM|82543GC]
                        [-cableconnected<1-N> on|off]
                        [-nictrace<1-N> on|off]
                        [-nictracefile<1-N> <filename>]
                        [-nicspeed<1-N> <kbps>]
                        [-hostifdev<1-N> none|<devicename>]
                        [-intnet<1-N> <network name>]
                        [-natnet<1-N> <network>|default]
                        [-macaddress<1-N> auto|<mac>]
                        [-uart<1-N> off|<I/O base> <IRQ>]
                        [-uartmode<1-N> disconnected|
                                        server <pipe>|
                                        client <pipe>|
                                        <devicename>]
                        [-gueststatisticsinterval <seconds>]
                        [-tapsetup<1-N> none|<application>]
                        [-tapterminate<1-N> none|<application>]
                        [-audio none|null|dsound|solaudio|oss|alsa|pulse|coreaudio]
                        [-audiocontroller ac97|sb16]
                        [-clipboard disabled|hosttoguest|guesttohost|
                                    bidirectional]
                        [-vrdp on|off]
                        [-vrdpport default|<port>]
                        [-vrdpaddress <host>]
                        [-vrdpauthtype null|external|guest]
                        [-vrdpmulticon on|off]
                        [-usb on|off]
                        [-usbehci on|off]
                        [-snapshotfolder default|<path>]

VBoxManage startvm      <uuid>|<name>
                        [-type gui|vrdp]

VBoxManage controlvm    <uuid>|<name>
                        pause|resume|reset|poweroff|savestate|
                        acpipowerbutton|acpisleepbutton|
                        keyboardputscancode <hex> [<hex> ...]|
                        setlinkstate<1-4> on|off |
```

```
                          usbattach <uuid>|<address> |
                          usbdetach <uuid>|<address> |
                          dvdattach none|<uuid>|<filename>|host:<drive> |
                          floppyattach none|<uuid>|<filename>|host:<drive> |
                          setvideomodehint <xres> <yres> <bpp> [display]|
                          setcredentials <username> <password> <domain>
                                        [-allowlocallogon <yes|no>]

VBoxManage discardstate   <uuid>|<name>

VBoxManage adoptstate     <uuid>|<name> <state_file>

VBoxManage snapshot       <uuid>|<name>
                          take <name> [-desc <desc>] |
                          discard <uuid>|<name> |
                          discardcurrent -state|-all |
                          edit <uuid>|<name>|-current
                               [-newname <name>]
                               [-newdesc <desc>] |
                          showvminfo <uuid>|<name>

VBoxManage registerimage  disk|dvd|floppy <filename>
                          [-type normal|immutable|writethrough] (disk only)

VBoxManage unregisterimage disk|dvd|floppy <uuid>|<filename>

VBoxManage showvdiinfo    <uuid>|<filename>

VBoxManage createvdi      -filename <filename>
                          -size <megabytes>
                          [-static]
                          [-comment <comment>]
                          [-register]
                          [-type normal|writethrough] (default: normal)

VBoxManage modifyvdi      <uuid>|<filename>
                          compact

VBoxManage clonevdi       <uuid>|<filename> <outputfile>

VBoxManage convertdd      <filename> <outputfile>
VBoxManage convertdd      stdin <outputfile> <bytes>

VBoxManage addiscsidisk   -server <name>|<ip>
                          -target <target>
                          [-port <port>]
                          [-lun <lun>]
                          [-encodedlun <lun>]
                          [-username <username>]
                          [-password <password>]
                          [-comment <comment>]

VBoxManage createhostif   <name>

VBoxManage removehostif   <uuid>|<name>
```

```
VBoxManage getextradata     global|<uuid>|<name>
                            <key>|enumerate

VBoxManage setextradata     global|<uuid>|<name>
                            <key>
                            [<value>] (no value deletes key)

VBoxManage setproperty      vdifolder default|<folder> |
                            machinefolder default|<folder> |
                            vrdpauthlibrary default|<library> |
                            websrvauthlibrary default|null|<library> |
                            hwvirtexenabled yes|no
                            loghistorycount <value>

VBoxManage usbfilter        add <index,0-N>
                            -target <uuid>|<name>|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>|<name>|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>|<name>|global

VBoxManage sharedfolder     add <vmname>|<uuid>
                            -name <name> -hostpath <hostpath>
                            [-transient] [-readonly]

VBoxManage sharedfolder     remove <vmname>|<uuid>
                            -name <name> [-transient]

VBoxManage vmstatistics     <vmname>|<uuid> [-reset]
                            [-pattern <pattern>] [-descriptions]
```

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.

8.1 VBoxManage list

The `list` command gives relevant information about your system and information about VirtualBox's current settings.

The following subcommands are available with `VBoxManage list`:

- `vms`, `hdds`, `dvds` and `floppies` all give you information about virtual machines and virtual disk images currently registered in VirtualBox, including all their settings, the unique identifiers (UUIDs) associated with them by VirtualBox and all files associated with them.
- `ostypes` lists all guest operating systems presently known to VirtualBox, along with the identifiers used to refer to them with the `modifyvm` command.
- `hostdvds`, `hostfloppies` and `hostifs`, respectively, list DVD, floppy and host networking interfaces on the host, along with the name used to access them from within VirtualBox.
- `hostusb` supplies information about USB devices attached to the host, notably information useful for constructing USB filters and whether they are currently in use by the host.
- `usbfilters` lists all global USB filters registered with VirtualBox – that is, filters for devices which are accessible to all virtual machines – and displays the filter parameters.
- `systemproperties` displays some global VirtualBox settings, such as minimum and maximum guest RAM and virtual hard disk size, folder settings and the current authentication library in use.

8.2 VBoxManage showvminfo

The `showvminfo` command shows information about a particular virtual machine. This is the same information as `VBoxManage list vms` would show for all virtual machines.

You will get information similar to the following:

```
$ VBoxManage showvminfo "Windows XP"
VirtualBox Command Line Management Interface Version 1.6.2
(C) 2005-2008 Sun Microsystems, Inc.
All rights reserved.
```

```
Name:            Windows XP
Guest OvS:        Other/Unknown
UUID:            1bf3464d-57c6-4d49-92a9-a5cc3816b7e7
Config file:     /home/username/.VirtualBox/Machines/Windows XP/Windows XP.xml
Memory size:     128MB
VRAM size:       8MB
Boot menu mode:  message and menu
ACPI:            on
IOAPIC:          off
Hardw. virt.ext: off
State:           powered off
Floppy:          empty
DVD:             empty
NIC 1:           disabled
NIC 2:           disabled
NIC 3:           disabled
NIC 4:           disabled
Audio:           disabled (Driver: Unknown)
VRDP:            disabled
USB:             disabled

USB Device Filters:
<none>

Shared folders:
<none>
```

8.3 VBoxManage registervm / unregistervm

The `registervm` command allows you to import a virtual machine definition in an XML file into VirtualBox. There are some restrictions here: the machine must not conflict with one already registered in 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` (see below), 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 then the XML definition file will be deleted.

8.4 VBoxManage createvm

This command creates a new XML virtual machine definition file.

The `-name <name>` parameter is required and must specify the name of the machine. Since this name is used by default as the file name of the settings file (with the extension `.xml`) and the machine folder (a subfolder of the `.VirtualBox/Machines` folder), it must conform to your host operating system's requirements for file name specifications. If the VM is later renamed, the file and folder names will change automatically.

However, if the `-basefolder <path>` and the `-settingsfile <filename>` options are used, the XML definition file will be given the name `<filename>` and the machine folder will be named `<path>`. In this case, the names of the file and the folder will not change if the virtual machine is renamed.

By default, this command only creates the XML file without automatically registering the VM with your VirtualBox installation. To register the VM instantly, use the optional `-register` option, or run `VBoxManage registervm` separately afterwards.

8.5 VBoxManage modifyvm

This command changes the properties of a registered virtual machine. Most of the properties that this command makes available correspond to the VM settings that VirtualBox graphical user interface displays in each VM's "Settings" dialog; these were described in chapter 3.7, *Virtual machine settings*, page 41. Some of the more advanced settings, however, are only available through the `VBoxManage` interface.

8.5.1 General settings

The following general settings are available through `VBoxManage modifyvm`:

- `-name <name>`: This changes the VM's name and possibly renames the internal virtual machine files, as described with `VBoxManage createvm` above.

- `-ostype <ostype>`: This specifies what guest operating system is supposed to run in the VM. As mentioned at chapter 3.2, *Creating a virtual machine*, page 28, this setting is presently purely descriptive. To learn about the various identifiers that can be used here, use `VBoxManage list ostypes`.

- `-memory <memorysize>`: This sets the amount of RAM, in MB, that the virtual machine should allocate for itself from the host. Again, see the remarks in chapter 3.2, *Creating a virtual machine*, page 28 for more information.

- `-vram <vramsize>`: This sets the amount of RAM that the virtual graphics card should have. See chapter 3.7.1, *General settings*, page 42 for details.

- `-acpi on|off`; `-ioapic on|off`: These two determine whether the VM should have ACPI and I/O APIC support, respectively; again, see chapter 3.7.1, *General settings*, page 42 for details.

- `-hwvirtex on|off|default`: This enables or disables the use of virtualization extensions in the processor of your host system.

- You can influence the BIOS logo that is displayed when a virtual machine starts up with a number of settings. Per default, a 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, if you are so inclined, replace the image that is shown, with your own logo. The image must be an uncompressed 256 color BMP file.

- `-biosbootmenu disabled|menuonly|messageandmenu`: This specifies whether the BIOS allows the user to select a temporary boot device. `menuonly` suppresses the message, but the user can still press F12 to select a temporary boot device.

- `-boot<1-4> none|floppy|dvd|disk|net`: This 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.

- `-snapshotfolder default|<path>`: This allows you to specify the folder in which snapshots will be kept for a virtual machine.

8.5.2 Storage settings

The following storage settings are available through `VBoxManage modifyvm`:

- `-hd<a|b|d> none|<uuid>|<filename>`: This specifies the settings for each of the three virtual hard disks that can be attached to a VM's IDE controller (primary master and slave, and secondary slave; the secondary master is always reserved for the virtual CD/DVD drive). For each of these three, specify either the UUID or a filename of a virtual disk that you have

  - either registered with `VBoxManage registerimage`; see chapter 8.10, *VBoxManage registerimage / unregisterimage*, page 109;
  - or created using `VBoxManage createvdi` with the `-register` option; see chapter 8.12, *VBoxManage createvdi*, page 109;
  - alternatively, specify the UUID of an iSCSI target that you have registered with `VBoxManage addiscsidisk`; see chapter 8.16, *VBoxManage addiscsidisk*, page 110.

- `-sata on|off`: this determines whether VirtualBox, in addition to the IDE controller, should also present an SATA controller as a second PCI device to the virtual machine. See chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62 for additional information.

- `-sataportcount <1-30>`: if the SATA controller is enabled, this determines how many ports the SATA controller should support.

- `-sataport<1-30> none|<uuid>|<filename>`: if the SATA controller is enabled, this specifies how an SATA slot should be occupied. This works just like the `-hd` options explained above.

- `-sataideemulation<1-4> <1-30>`: if the SATA controller is enabled, this specifies which SATA ports should operate in IDE emulation mode. As explained in chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62, by default, this is the case for SATA ports 1-4; with this command, you can map four IDE channels to any of the 30 supported SATA ports.

- `-dvd none|<uuid>|<filename>|host:<drive>`: This specifies what VirtualBox should provide to the VM as the virtual CD/DVD drive; specify either the UUID or the filename of an image file that you have registered with `VBoxManage registerimage` (see chapter 8.10, *VBoxManage registerimage / unregisterimage*, page 109). Alternatively, specify "host:" with the drive specification of your host's drive.

- `-dvdpassthrough on|off`: With this, you can enable DVD writing support (currently experimental; see chapter 3.7.3, *CD/DVD-ROM and floppy settings*, page 46).

- `-floppy disabled|empty|<uuid>|<filename>|host:<drive>`: This is the floppy equivalent to the `-dvd` option described above. `disabled` completely disables the floppy controller, whereas `empty` keeps the floppy controller enabled, but without a media inserted.

8.5.3 Networking settings

The following networking settings are available through `VBoxManage modifyvm`:

- `-nic<1-N> none|null|nat|hostif|intnet`: With this, you can set, for each of the VM's virtual network cards, what type of networking should be available. They can be not present (none), not connected to the host (null), use network address translation (nat), a host interface (hostif) or communicate with other virtual machines using internal networking (intnet). These options correspond to the modes which are described in detail in chapter 6.2, *Introduction to networking modes*, page 69.

- `-nictype<1-N> Am79C970A|Am79C973|82540EM`: This allows you, for each of the VM's virtual network cards, to specify which networking hardware VirtualBox presents to the guest; see chapter 6.1, *Virtual networking hardware*, page 69.

- `-cableconnected<1-N> on|off`: This allows 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 resetting certain software components in the VM.

- With the "nictrace" options, you can optionally trace network traffic, 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>` what file the trace should be logged to.

- `-hostifdev<1-N> none|<devicename>`: If host interface networking has been enabled for a virtual network card (see the `-nic` option above; otherwise this setting has no effect), use this option to specify which host interface the given virtual network interface will use.

  For Windows hosts, this should be the name of a VirtualBox host interface which you have created using the `createhostif` command. For Linux hosts, this should be the name of an existing static interface or `none` if you wish to allocate an interface dynamically. In the latter case, you should also specify the creation and termination scripts for the interface with `-tapsetup<1-4>` and `-tapterminate<1-4>`. For details, please see chapter 6.5, *Introduction to Host Interface Networking (HIF)*, page 73.

- `-intnet<1-N> network`: If internal networking has been enabled for a virtual network card (see the `-nic` option above; otherwise this setting has no effect), use this option to specify the name of the internal network (see chapter 6.9, *Internal networking*, page 85).

- `-macaddress<1-N> auto|<mac>`: With this option you can set the MAC address of the virtual network card. Per default, each virtual network card is assigned a random address by VirtualBox at VM creation.

8.5.4 Serial port, audio, clipboard, VRDP and USB settings

The following other settings are available through `VBoxManage modifyvm`:

- `-uart<1-N> off|<I/O base> <IRQ>`: With this option you can configure virtual serial ports for the VM; see chapter 3.7.9, *Serial ports*, page 50 for an introduction.

- `-uartmode<1-N> <arg>`: This setting controls how VirtualBox connects a given virtual serial port (previously configured with the `-uartX` setting, see above) to the host on which the virtual machine is running. As described in detail in chapter 3.7.9, *Serial ports*, page 50, 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" – like a real COM port without a cable.
  - `server <pipename>`: On a Windows host, this tells 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 begin with `\\.\pipe\`.

    On a Linux host, instead of a named pipe, a local domain socket is used.
  - `client <pipename>`: This operates just like `server ...`, except that the pipe (or local domain socket) is not created by VirtualBox, but assumed to exist already.
  - `<devicename>`: If, instead of the above, 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 look like `/dev/ttyS0`. This allows you to "wire" a real serial port to a virtual machine.

- `-audio none|null|oss`: With this option, you can set whether the VM should have audio support.

- `-clipboard disabled|hosttoguest|guesttohost| bidirectional`: With this setting, you can select whether the guest operating system's clipboard should be shared with the host; see chapter 3.7.1, *General settings*, page 42. This requires that the Guest Additions be installed in the virtual machine.

- `-vrdp on|off`: With the VirtualBox graphical user interface, this enables or disables the built-in VRDP server. Note that if you are using `VBoxHeadless` (see chapter 7.4.1, *VBoxHeadless, the VRDP-only server*, page 91), VRDP output is always enabled.

- `-vrdpport default|<port>`: This lets you specify which port should be used; "default" or "0" means port 3389, the standard port for RDP. Only one machine can use a given port at a time.

- `-vrdpauthtype null|external|guest`: This allows you to choose whether and how authorization will be performed; see chapter 7.4.4, *RDP authentication*, page 93 for details.

- `-usb on|off`: This option enables or disables the VM's virtual USB controller; see chapter 3.7.6.1, *USB settings*, page 48 for details.

- `-usbehci on|off`: This option enables or disables the VM's virtual USB 2.0 controller; see chapter 3.7.6.1, *USB settings*, page 48 for details.

8.6 VBoxManage startvm

This command starts a virtual machine that is currently in the "Powered off" or "Saved" states. This is provided for backwards compatibility only.

The optional `-type` specifier determines whether the machine will be started in a window (GUI mode, which is the default) or whether the output should go through `VBoxHeadless`, the VRDP-only server; see chapter 7.4.1, *VBoxHeadless, the VRDP-only server*, page 91 for more information.

Note: We recommend to start virtual machines directly by running the respective front-end, as you might otherwise miss important error and state information that VirtualBox may display on the console. This is especially important for front-ends other than `VirtualBox`, our graphical user interface, because those cannot display error messages in a popup window. See chapter 7.4.1, *VBoxHeadless, the VRDP-only server*, page 91 for more information.

8.7 VBoxManage controlvm

The `controlvm` subcommand allows 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 changing its state for good. The VM window will be painted in gray to indicate that the VM is currently paused. (This is equivalent to selecting the "Pause" item in the "VM" menu of the GUI.)

- Use `VBoxManage controlvm <vm> resume` to undo a previous `pause` command. (This is equivalent to selecting the "Resume" item in the "VM" 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, which will restart and boot the guest operating system again immediately. The state of the VM is not saved beforehand, and data may be lost. (This is equivalent to selecting the "Reset" item in the "VM" 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. Again, the state of the VM is not saved beforehand, and data may be lost. (This is equivalent to

selecting the “Close” item in the “VM” menu of the GUI or pressing the window’s close button, and then selecting “Power off the machine” in the dialog.)

After this, the VM’s state will be “Powered off”. From there, it can be started again; see chapter 8.6, *VBoxManage startvm*, page 107.

- `VBoxManage controlvm <vm> savestate` will save the current state of the VM to disk and then stop the VM. (This is equivalent to selecting the “Close” item in the “VM” menu of the GUI or pressing the window’s close button, and then selecting “Save the machine state” in the dialog.)

  After this, the VM’s state will be “Saved”. From there, it can be started again; see chapter 8.6, *VBoxManage startvm*, page 107.

A few extra options are available with `controlvm` that do not directly affect the VM’s running state:

- The `setlinkstate<1-4>` operation connects or disconnects virtual network cables from their network interfaces
- `usbattach` and `usbdettach` make host USB devices visible 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.

  You can use `VBoxManage list usbhost` to locate this information.
- `dvdattach` inserts a DVD image into the virtual machine or connects it to the host DVD drive. With this command (as opposed to `VBoxManage modifyvm`), the image file does not first have to be registered with VirtualBox.

  You can use `VBoxManage list hostdvds` to display all the drives found on the host and the names VirtualBox uses to access them.
- `floppyattach` works in a similar way.
- `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.
- The `setcredentials` operation is used for remote logons in Windows guests. For details, please refer to chapter 9.2, *Automated Windows guest logons (VBoxGINA)*, page 115.

8.8 VBoxManage discardstate

This command discards the saved state of a virtual machine which is not currently running, which will cause its 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.

8.9 VBoxManage snapshot

This command is used for taking snapshots of a virtual machine and for manipulating and discarding snapshots.

The `take` operation takes a snapshot of a virtual machine. You must supply a name for the snapshot and can optionally supply a description.

The `discard` operation discards a snapshot specified by name or by identifier (UUID).

The `discardcurrent` operation will either revert the current state to the most recent snapshot (if you specify the `-state` option) or discard the last snapshot and revert to the last but one (with the `-all` option).

8.10 VBoxManage registerimage / unregisterimage

These commands register or unregister hard disk, DVD or floppy images in VirtualBox. This is the command-line equivalent of the Virtual Disk Manager; see chapter 3.5, *The Virtual Disk Manager*, page 39 for more information.

Note however that when you unregister a hard disk image using VBoxManage, it will not be deleted from the host computer's hard drive.

8.11 VBoxManage showvdiinfo

This command shows information about a virtual hard disk image, notably its size, its size on disk, its type and the VM it is in use by.

8.12 VBoxManage createvdi

This command creates a new virtual hard disk image. You must specify the filename for the new image and the virtual size. If you give the `-static` option, disk space for the whole image will be allocated at once on the host. With the `-comment` option you can attach a comment to the image. The `-register` option, if given, tells VirtualBox to register the image for use with virtual machines.

You can use the `-type` option to create a disk in write-through mode, which will not be affected by snapshots; see chapter 5.2, *Virtual Disk Image (VDI) files*, page 64 for details. (As described there, you cannot *create* a VDI with the "immutable" type, as it would then always remain empty.)

8.13 VBoxManage modifyvdi

The `modifyvdi` command can be used to compact disk images, i.e. remove blocks that only contains zeroes. For this operation to be effective, it is required to zero out free space in the guest system using a suitable software tool.

8.14 VBoxManage clonevdi

This command duplicates a registered virtual hard disk image to a new image file with a new unique identifier (UUID). The new image can be transferred to another host system or imported into VirtualBox again using the Virtual Disk Manager; see chapter 3.5, *The Virtual Disk Manager*, page 39 and chapter 5.3, *Cloning disk images*, page 65.

8.15 VBoxManage convertdd

This command converts a raw disk image to a VirtualBox Disk Image (VDI) file. The syntax is as follows:

```
VBoxManage convertdd        <filename> <outputfile>
VBoxManage convertdd        stdin <outputfile> <bytes>
```

8.16 VBoxManage addiscsidisk

The `addiscsidisk` command attaches an iSCSI network storage unit to VirtualBox. The iSCSI target can then be made available to and used by a virtual machine as though it were a standard write-through virtual disk image.

This command has the following syntax:

```
VBoxManage addiscsidisk -server <name>|<ip>
                        -target <target>
                        [-port <port>]
                        [-lun <lun>]
                        [-username <username>]
                        [-password <password>]
                        [-comment <comment>]
```

where the parameters mean:

server The host name or IP address of the iSCSI target.

target Target name string. This is determined by the iSCSI target and used to identify the storage resource.

port 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.

username, password Username and password for target authentication, if required (optional).

> **Note:** Currently, username and password are stored without encryption (i.e. in cleartext) in the machine configuration file.

comment Any description that you want to have stored with this item (optional; e.g. "Big storage server downstairs"). This is stored internally only and not needed for operation.

8.17 VBoxManage createhostif/removehostif

These two commands add and remove, respectively, virtual network interfaces on Windows hosts. See chapter 6.6, *Host Interface Networking and bridging on Windows hosts*, page 74 for details.

8.18 VBoxManage getextradata/setextradata

These commands let you attach and retrieve string data to a virtual machine or to a VirtualBox configuration (by specifying `global` instead of a virtual machine name). You must specify a key (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
```

would associate the string "2006.01.01" with the key installdate for the virtual machine Fedora5, and "2006.02.02" on the machine SUSE10. You could retrieve the information as follows:

```
VBoxManage getextradata Fedora5 installdate
```

which would return

```
VirtualBox Command Line Management Interface Version 1.6.2
(C) 2005-2008 Sun Microsystems, Inc.
All rights reserved.

Value: 2006.01.01
```

8.19 VBoxManage setproperty

This command is used to change global settings which affect the entire VirtualBox installation. Some of these correspond to the settings in the "Global settings" dialog in the graphical user interface. The following properties are available:

vdifolder This specifies the default folder that is used to keep Virtual Disk Image (VDI) files.

machinefolder This specifies the default folder in which virtual machine definitions are kept; see chapter 9.1, *VirtualBox configuration data*, page 114 for details.

vrdpauthlibrary This specifies which library to use when "external" VRDP authentication has been selected for a particular virtual machine; see chapter 7.4.4, *RDP authentication*, page 93 for details.

websrvauthlibrary This specifies which library the webservice uses to authenticate users; see chapter 10.1.1, *Introduction*, page 128 for details.

hwvirtexenabled This selects whether or not hardware virtualization support is enabled by default. Note: This feature may still be experimental at the time you read this.

8.20 VBoxManage usbfilter add/modify/remove

The `usbfilter` commands are used for working with USB filters in virtual machines, or global filters which affect the whole 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. So for example, if the first global filter says to hold (make available) a particular Kingston memory stick device and the second to ignore all Kingston devices, that memory stick will be available to any machine with an 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 and for global filters, `action` says whether to allow machines 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, and for local filters, whether they are for local devices, remote (over an RDP connection) or either.

When you modify a USB filter using `usbfilter modify`, you must specify the filter by index (see the output of `VBoxManage list usbfilters` to find global filter indexes and that of `VBoxManage showvminfo` to find indexes for individual machines) and by target, which is either a virtual machine or "global". 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.

8.21 VBoxManage sharedfolder add/remove

This command allows you to share folders on the host computer with guest operating systems. For this, the guest systems must have a version of the VirtualBox guest additions installed which supports this functionality.

Shared folders are described in detail in chapter 4.5, *Folder sharing*, page 59.

8.22 VBoxManage updatesettings

The `updatesettings` command updates all VirtualBox configuration files from an earlier to the current version. You will need this when you upgrade your version of VirtualBox, but should not need it apart from that.

9 Advanced topics

9.1 VirtualBox configuration data

For each system user, VirtualBox stores configuration data in the user's home directory, as per the conventions of the host operating system:

- On Windows, this is `%HOMEDRIVE%%HOMEPATH%\.VirtualBox`; typically something like `C:\Documents and Settings\Username\.VirtualBox`.
- On Unix-like systems (Linux, Solaris), this is `$HOME/.VirtualBox`.
- On Mac OS X, this is `$HOME/Library/VirtualBox`.

VirtualBox creates this configuration directory automatically, if necessary. Optionally, you can supply an alternate configuration directory by setting the `VBOX_USER_HOME` environment variable.

VirtualBox stores all its global and machine-specific configuration data in XML documents. We intentionally do not document the specifications of these files, as we must reserve the right to modify them in the future. We therefore strongly suggest that you do not edit these files manually. VirtualBox provides complete access to its configuration data through its the `VBoxManage` command line tool (see chapter 8, *VBoxManage reference*, page 96) and its API (see chapter 10, *VirtualBox programming interfaces*, page 127).

In the configuration directory, `VirtualBox.xml` is the main configuration file. This includes global configuration options and the media and virtual machine registry. The media registry links to all CD/DVD, floppy and disk images that have been added to the Virtual Disk Manager. For each registered VM, there is one entry which points to the VM configuration file, also in XML format.

You can globally change some of the locations where VirtualBox keeps extra configuration and data by selecting "Global settings" from the "File" menu in the VirtualBox main window. Then, in the window that pops up, click on the "General" tab.

- Virtual machine settings and files are, by default, saved as XML files in a subdirectory of the `.VirtualBox/Machines` directory. You can change the location of this main "Machines" folder in the "Global settings" dialog.

  By default, for each virtual machine, VirtualBox uses another subdirectory of the "Machines" directory that carries the same name as the virtual machine. As a result, your virtual machine names must conform to the conventions of your operating system for valid file names. For example, a

machine called "Fedora 6" would, by default, have its settings saved in `.VirtualBox/Machines/Fedora 6/Fedora 6.xml`. If you would like more control over the file names used, you can create the machine using `VBoxManage createvm` with the `-settingsfile` option; see chapter 8.4, *VBoxManage createvm*, page 101.

The virtual machine directory will be renamed if you change the machine name. If you do not wish this to happen, you can create the machine using `VBoxManage createvm` with the `-basefolder` option. In this case, the folder name will never change.

- VirtualBox keeps snapshots and saved states in another special folder for each virtual machine. By default, this is a subfolder of the virtual machine folder called `Snapshots` – in our example, `.VirtualBox/Machines/Fedora 6/Snapshots`. You can change this setting for each machine using `VBoxManage` as well.
- VDI container files are, by default, created in the `.VirtualBox/VDI` directory. In particular, this directory is used when the "Create new virtual disk" wizard is started to create a new VDI file. Changing this default is probably most useful if the disk containing your home directory does not have enough room to hold your VDI files, which can grow very large.

9.2 Automated Windows guest logons (VBoxGINA)

When Windows is running in a virtual machine, it might be desirable to perform coordinated and automated logons of guest operating systems using credentials from a master logon system. (With "credentials", we are referring to logon information consisting of user name, password and domain name, where each value might be empty.) Since Windows NT, Windows has provided a modular system logon subsystem ("Winlogon") which can be customized and extended by means of so-called GINA modules (Graphical Identification and Authentication). The VirtualBox Guest Additions for Windows come with such a GINA module and therefore allow Windows guests to perform automated logons.

To activate the GINA module, first install the Guest Additions. You will then find the GINA module – a file called `VBoxGINA.dll` – in the Additions target directory. Copy this file to the Windows `SYSTEM32` directory. Then, in the registry, create the following key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
                      Winlogon\GinaDLL
```

with a value of `VBoxGINA.dll`.

Note: The VirtualBox GINA is implemented as a wrapper around the standard Windows GINA (`MSGINA.DLL`) so it will most likely not work correctly with 3rd party GINA modules.

To set credentials, use the following command on a *running* VM:

```
VBoxManage controlvm "Windows XP" \
      setcredentials "John Doe" "secretpassword" "DOMTEST"
```

While the VM is running, the credentials can be queried by the VirtualBox GINA module using the VirtualBox Guest Additions device driver. When Windows is in "logged out" mode, the GINA module will constantly poll for credentials and if they are present, a logon will be attempted. After retrieving the credentials, the GINA module will erase them so that the above command will have to be repeated for subsequent logons.

For security reasons, credentials are not stored in any persistent manner and will be lost when the VM is reset. Also, the credentials are "write-only", i.e. there is no way to retrieve the credentials from the host side. Credentials can be reset from the host side by setting empty values.

For Windows XP guests, the logon subsystem has to be configured to use the classical logon dialog as the VirtualBox GINA does not support the XP style welcome dialog.

The following command forces VirtualBox to keep the credentials after they were read by the guest and on VM reset:

```
VBoxManage setextradata "Windows XP" \
      VBoxInternal/Devices/VMMDev/0/Config/KeepCredentials 1
```

Note that this is a potential security risk as a malicious application running on the guest could request this information using the proper interface.

9.3 Custom external VRDP authentication

As described in chapter 7.4.4, *RDP authentication*, page 93, VirtualBox supports arbitrary external modules to perform authentication with its VRDP servers. When the authentication method is set to "external" for a particular VM, VirtualBox calls the library that was specified with `VBoxManage setproperty vrdpauthlibrary`. This library will be loaded by the VM process on demand, i.e. when the first RDP connection is made by an external client.

External authentication is the most flexible as the external handler can both choose to grant access to everyone (like the "null" authentication method would) and delegate the request to the guest authentication component. When delegating the request to the guest component, it will still be called afterwards with the option to override the result.

A VRDP authentication library is required to implement exactly one entry point:

```
#include "VRDPAuth.h"

/**
 * Authentication library entry point. Decides whether to allow
 * a client connection.
 *
```

```
 * Parameters:
 *
 *   pUuid            Pointer to the UUID of the virtual machine
 *                    which the client connected to.
 *   guestJudgement   Result of the guest authentication.
 *   szUser           User name passed in by the client (UTF8).
 *   szPassword       Password passed in by the client (UTF8).
 *   szDomain         Domain passed in by the client (UTF8).
 *
 * Return code:
 *
 *   VRDPAuthAccessDenied    Client access has been denied.
 *   VRDPAuthAccessGranted   Client has the right to use the
 *                           virtual machine.
 *   VRDPAuthDelegateToGuest Guest operating system must
 *                           authenticate the client and the
 *                           library must be called again with
 *                           the result of the guest
 *                           authentication.
 */
VRDPAuthResult VRDPAUTHCALL VRDPAuth(
    PVRDPAUTHUUID pUuid,
    VRDPAuthGuestJudgement guestJudgement,
    const char *szUser,
    const char *szPassword
    const char *szDomain)
{
    /* process request against your authentication source of choice */
    return VRDPAuthAccessGranted;
}
```

The second arguments contains information about the guest authentication status. For the first call, it is always set to `VRDPAuthGuestNotAsked`. In case the function returns `VRDPAuthDelegateToGuest`, a guest authentication will be attempted and another call to the method is made with its result. This can be either granted / denied or no judgement (the guest component chose for whatever reason to not make a decision). In case there is a problem with the guest authentication module (e.g. the Additions are not installed or not running or the guest did not respond within a timeout), the “not reacted” status will be returned.

9.4 Secure labeling with VBoxSDL

When running guest operating systems in fullscreen mode, the guest operating system usually has control over the whole screen. This could present a security risk as the guest operating system might fool the user into thinking that it is either a different system (which might have a higher security level) or it might present messages on the screen that appear to stem from the host operating system.

In order to protect the user against the above mentioned security risks, the secure labeling feature has been developed. Secure labeling is currently available only for VBoxSDL. When enabled, a portion of the display area is reserved for a label in which

a user defined message is displayed. The label height in set to 20 pixels in VBoxSDL. The label font color and background color can be optionally set as hexadecimal RBG color values. The following syntax is used to enable secure labeling:

```
VBoxSDL -securelabel -seclabelfnt ~/fonts/arial.ttf \
        -seclabelsiz 14 "Windows XP" \
        -seclabelfbcol 00FF00 -seclabelbgcol 00FFFF
```

In addition to enabling secure labeling, a TrueType font has to be supplied.

Typically, full screen resolutions are limited to certain “standard” geometries such as 1024 x 768. Increasing this by twenty lines is not usually feasible, so in most cases, VBoxSDL will chose the next higher resolution, e.g. 1280 x 1024 and the guest’s screen will not cover the whole display surface. If VBoxSDL is unable to choose a higher resolution, the secure label will be painted on top of the guest’s screen surface. In order to address the problem of the bottom part of the guest screen being hidden, VBoxSDL can provide custom video modes to the guest that are reduced by the height of the label. For Windows guests and recent Solaris and Linux guests, the VirtualBox Guest Additions automatically provide the reduced video modes. Additionally, the VESA BIOS has been adjusted to duplicate its standard mode table with adjusted resolutions. The adjusted mode IDs can be calculated using the following formula:

```
reduced_modeid = modeid + 0x30
```

For example, in order to start Linux with 1024 x 748 x 16, the standard mode 0x117 (1024 x 768 x 16) is used as a base. The Linux video mode kernel parameter can then be calculated using:

```
vga = 0x200 | 0x117 + 0x30
vga = 839
```

The reason for duplicating the standard modes instead of only supplying the adjusted modes is that most guest operating systems require the standard VESA modes to be fixed and refuse to start with different modes.

When using the X.org VESA driver, custom modelines have to be calculated and added to the configuration (usually in /etc/X11/xorg.conf. A handy tool to determine modeline entries can be found at http://www.tkk.fi/Misc/Electronics/faq/vga2rgb/calc.html.)

9.5 Custom VESA resolutions

Apart from the standard VESA resolutions, the VirtualBox VESA BIOS allows you to add up to 16 custom video modes which will be reported to the guest operating system. When using Windows guests with the VirtualBox Guest Additions, a custom graphics driver will be used instead of the fallback VESA solution so this information does not apply.

Additional video modes can be configured for each VM using the extra data facility. The extra data key is called CustomVideoMode<x> with x being a number from 1 to

16. Please note that modes will be read from 1 until either the following number is not defined or 16 is reached. The following example adds a video mode that corresponds to the native display resolution of many notebook computers:

```
VBoxManage setextradata "Windows XP" \
          "CustomVideoMode1" "1400x1050x16"
```

The VESA mode IDs for custom video modes start at 0x160. In order to use the above defined custom video mode, the following command line has be supplied to Linux:

```
vga = 0x200 | 0x160
vga = 864
```

For guest operating systems with VirtualBox Guest Additions, a custom video mode can be set using the video mode hint feature.

9.6 Multiple monitors for the guest

VirtualBox allows the guest to use multiple virtual monitors. Up to sixty-four virtual monitors are supported.

Note:

1. Multiple monitors currently work only with Windows XP guests, and Guest Additions must be installed, as the implementation resides in the Guest Additions video driver.
2. Multiple monitors work only with the VBoxHeadless frontend. You must also enable VRDP multiconnection mode (see chapter 7.4.6, *VRDP multiple connections*, page 94) to access two or more VM displays when the guest is using multiple monitors.
3. The guest video RAM size should be increased when multiple monitors are used. The VRAM is shared among the virtual monitors so that only part of it is available for each one. Therefore the available resolutions and color depths will be reduced if the VRAM size remains the same and multiple monitors are enabled.

The following command enables three virtual monitors for the VM:

```
VBoxManage modifyvm VMNAME -monitorcount 3
```

The RDP client can select the virtual monitor number to connect to using the `domain` logon parameter. If the parameter ends with `@` followed by a number, VBoxHeadless interprets this number as the screen index. The primary guest screen is selected with `@1`, the first secondary screen is `@2`, etc.

The MS RDP6 client does not let you specify a separate domain name. Instead, use `domain\username` in the `Username:` field – for example, `@2\name`. `name` must be supplied, and must be the name used to log in if the VRDP server is set up to require credentials. If it is not, you may use any text as the username.

9.7 Releasing modifiers with VBoxSDL on Linux

When switching from a X virtual terminal (VT) to another VT using Ctrl-Alt-Fx while the VBoxSDL window has the input focus, the guest will receive Ctrl and Alt keypress events without receiving the corresponding key release events. This is an architectural limitation of Linux. In order to reset the modifier keys, it is possible to send `SIGUSR1` to the VBoxSDL main thread (first entry in the `ps` list). For example, when switching away to another VT and saving the virtual machine from this terminal, the following sequence can be used to make sure the VM is not saved with stuck modifiers:

```
kill -usr1 <pid>
./VBoxManage controlvm "Windows 2000" savestate
```

9.8 Using serial ports

Starting with version 1.4, VirtualBox provided support for virtual serial ports, which, at the time, was rather complicated to set up with a sequence of `VBoxManage setextradata` statements. Since version 1.5, that way of setting up serial ports is no longer necessary and *deprecated*. To set up virtual serial ports, use the methods now described in chapter 3.7.9, *Serial ports*, page 50.

> **Note:** For backwards compatibility, the old `setextradata` statements, whose description is retained below from the old version of the manual, take *precedence* over the new way of configuring serial ports. As a result, if configuring serial ports the new way doesn't work, make sure the VM in question does not have old configuration data such as below still active.

The old sequence of configuring a serial port used the following 6 commands:

```
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/serial/0/Config/IRQ" 4
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/serial/0/Config/IOBase" 0x3f8
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/serial/0/LUN#0/Driver" Char
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/serial/0/LUN#0/AttachedDriver/Driver" NamedPipe
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/serial/0/LUN#0/AttachedDriver/Config/Location"
   "\\.\pipe\vboxCOM1"
```

```
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/serial/0/LUN#0/AttachedDriver/Config/IsServer"
   1
```

This sets up a serial port in the guest with the default settings for COM1 (IRQ 4, I/O address 0x3f8) and the `Location` setting assumes that this configuration is used on a Windows host, because the Windows named pipe syntax is used. Keep in mind that on Windows hosts a named pipe must always start with `\\.\pipe\`". On Linux the same config settings apply, except that the path name for the `Location` can be chosen more freely. Local domain sockets can be placed anywhere, provided the user running VirtualBox has the permission to create a new file in the directory. The final command above defines that VirtualBox acts as a server, i.e. it creates the named pipe itself instead of connecting to an already existing one.

9.9 Using a raw host hard disk from a guest

Starting with version 1.4, as an alternative to using virtual disk images (as described in detail in chapter 5, *Virtual storage*, page 62), VirtualBox can also present either entire physical hard disks or selected partitions thereof as virtual disks to virtual machines.

With VirtualBox, this type of access is called "raw hard disk access"; it allows a guest operating system to access its virtual hard disk much more quickly than with disk images, since data does not have to pass through two file systems (the one in the guest and the one on the host).

> **Warning:** Raw hard disk access is for expert users only. Incorrect use or use of an outdated configuration can lead to **total loss of data** on the physical disk. Most importantly, *do not* attempt to boot the partition with the currently running host operating system in a guest. This will lead to severe data corruption.

Raw hard disk access – both for entire disks and individual partitions – is implemented as part of the VMDK image format support (see chapter 5.4, *VMDK image files*, page 66). As a result, you will need to create a special VMDK image file which defines where the data will be stored. After creating such a special VMDK image, you can use it like a regular virtual disk image. For example, you can use the Virtual Disk Manager (chapter 3.5, *The Virtual Disk Manager*, page 39) or `VBoxManage` to assign the image to a virtual machine.

9.9.1 Access to entire physical hard disk

While this variant is the simplest to set up, you must be aware that this will give a guest operating system direct and full access to an *entire physical disk*. If your *host* operating system is also booted from this disk, please take special care to not access the partition from the guest at all. On the positive side, the physical disk can be repartitioned in

arbitrary ways without having to recreate the image file that gives access to the raw disk.

To create an image that represents an entire physical hard disk (which will not contain any actual data, as this will all be stored on the physical disk), on a Linux host, use the command

```
VBoxManage internalcommands createrawvmdk -filename /path/to/file.vmdk
        -rawdisk /dev/sda
```

This creates the image `/path/to/file.vmdk` (must be absolute), and all data will be read and written from `/dev/sda`.

On a Windows host, instead of the above device specification, use e.g. `\\.\PhysicalDrive0`.

Creating the image requires read/write access for the given device. Read/write access is also later needed when using the image from a virtual machine.

Just like with regular disk images, this does not automatically register the newly created image in the internal registry of hard disks. If you want this done automatically, add `-register`:

```
VBoxManage internalcommands createrawvmdk -filename /path/to/file.vmdk
        -rawdisk /dev/sda -register
```

After registering, you can assign the newly created image to a virtual machine with

```
VBoxManage modifyvm WindowsXP -hda /path/to/file.vmdk
```

When this is done the selected virtual machine will boot from the specified physical disk.

9.9.2 Access to individual physical hard disk partitions

This “raw partition support” is quite similar to the “full hard disk” access described above. However, in this case, any partitioning information will be stored inside the VMDK image, so you can e.g. install a different boot loader in the virtual hard disk without affecting the host’s partitioning information. While the guest will be able to *see* all partitions that exist on the physical disk, access will be filtered in that reading from partitions for which no access is allowed the partitions will only yield zeroes, and all writes to them are ignored.

To create a special image for raw partition support (which will contain a small amount of data, as already mentioned), on a Linux host, use the command

```
VBoxManage internalcommands createrawvmdk -filename /path/to/file.vmdk
        -rawdisk /dev/sda -partitions 1,5
```

As you can see, the command is identical to the one for “full hard disk” access, except for the additional `-partitions` parameter. This example would create the image `/path/to/file.vmdk` (which, again, must be absolute), and partitions 1 and 5 of `/dev/sda` would be made accessible to the guest.

VirtualBox uses the same partition numbering as your Linux host. As a result, the numbers given in the above example would refer to the first primary partition and the first logical drive in the extended partition, respectively.

On a Windows host, instead of the above device specification, use e.g. `\\.\PhysicalDrive0`. Partition numbers are the same on Linux and Windows hosts.

The numbers for the list of partitions can be taken from the output of

```
VBoxManage internalcommands listpartitions -rawdisk /dev/sda
```

The output lists the partition types and sizes to give the user enough information to identify the partitions necessary for the guest.

Images which give access to individual partitions are specific to a particular host disk setup. You cannot transfer these images to another host; also, whenever the host partitioning changes, the image *must be recreated*.

Creating the image requires read/write access for the given device. Read/write access is also later needed when using the image from a virtual machine. If this is not feasible, there is a special variant for raw partition access (currently only available on Linux hosts) that avoids having to give the current user access to the entire disk. To set up such an image, use

```
VBoxManage internalcommands createrawvmdk -filename /path/to/file.vmdk
        -rawdisk /dev/sda -partitions 1,5 -relative
```

When used from a virtual machine, the image will then refer not to the entire disk, but only to the individual partitions (in the example `/dev/sda1` and `/dev/sda5`). As a consequence, read/write access is only required for the affected partitions, not for the entire disk. During creation however, read-only access to the entire disk is required to obtain the partitioning information.

In some configurations it may be necessary to change the MBR code of the created image, e.g. to replace the Linux boot loader that is used on the host by another boot loader. This allows e.g. the guest to boot directly to Windows, while the host boots Linux from the "same" disk. For this purpose the `-mbr` parameter is provided. It specifies a file name from which to take the MBR code. The partition table is not modified at all, so a MBR file from a system with totally different partitioning can be used. An example of this is

```
VBoxManage internalcommands createrawvmdk -filename /path/to/file.vmdk
        -rawdisk /dev/sda -partitions 1,5 -mbr winxp.mbr
```

The modified MBR will be stored inside the image, not on the host disk.

For each of the above variants, you can register the resulting image for immediate use in VirtualBox by adding `-register` to the respective command line. The image will then immediately appear in the list of registered disk images. An example is

```
VBoxManage internalcommands createrawvmdk -filename /path/to/file.vmdk
        -rawdisk /dev/sda -partitions 1,5 -relative -register
```

which creates an image referring to individual partitions, and registers it when the image is successfully created.

9.10 Allowing a virtual machine to start even with unavailable CD/DVD/floppy devices

When, on VM startup, a CD, DVD or floppy device is unavailable, VirtualBox by default prints an error message and refuses to start the virtual machine. In some situations this behavior is not desirable.

The behavior can be changed for the CD/DVD drive with the following configuration change command:

```
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/piix3ide/0/LUN#2/Config/AttachFailError" 0
```

The equivalent command for the floppy drive is:

```
VBoxManage setextradata "YourVM"
   "VBoxInternal/Devices/i82078/0/LUN#0/Config/AttachFailError" 0
```

You will still get a warning message that a device is not available. Some guest operating systems may show strange behavior when using saved state or snapshots, especially if a previously mounted medium is no longer available when the virtual machine is resumed.

9.11 Configuring the address of a NAT network interface

In NAT mode, the guest network interface is assigned to the IPv4 range `10.0.x.0/24` by default where `x` corresponds to the instance of the NAT interface +2 of that VM. So `x` is 2 if there is only once NAT instance active. In that case the guest is assigned to the address `10.0.2.15`, the gateway is set to `10.0.2.2` and the name server can be found at `10.0.2.3`.

If, for any reason, the NAT network needs to be changed, this can be achieved with the following command:

```
VBoxManage modifyvm "My VM" -natnet1 "192.168/16"
```

This command would reserve the network addresses `192.168.0.0` ... `192.168.254.254` for the first NAT network instance of “My VM”. The guest IP would be assigned to `192.168.0.15` and the default gateway could be found at `192.168.0.2`.

9.12 Configuring the maximum resolution of guests when using the graphical frontend

When guest systems with the Guest Additions installed are started using the graphical frontend (the normal VirtualBox application), they will not be allowed to use screen

resolutions greater than the host's screen size unless the user manually resizes them by dragging the window, switching to fullscreen or seamless mode or sending a video mode hint using VBoxManage. This behavior is what most users will want, but if you have different needs, it is possible to change it by issuing one of the following commands from the command line:

```
VBoxManage setextradata global GUI/MaxGuestResolution any
```

will remove all limits on guest resolutions.

```
VBoxManage setextradata global GUI/MaxGuestResolution
    >width,height<
```

manually specifies a maximum resolution.

```
VBoxManage setextradata global GUI/MaxGuestResolution auto
```

restores the default settings. Note that these settings apply globally to all guest systems, not just to a single machine.

9.13 Configuring the BIOS DMI information

The DMI data VirtualBox provides to guests can be changed for a specific VM. Use the following commands to configure the DMI BIOS information:

```
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVendor" \
   "Host BIOS Vendor"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVersion" \
   "Host BIOS Version"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseDate" \
   "Host BIOS Release Date"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMajor" \
   1
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMinor" \
   2
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMajor" \
   3
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMinor" \
   4
```

Use the following commands to configure the DMI System information:

```
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiSystemVendor" \
   "Host System Vendor"
```

```
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiSystemProduct" \
   "Host System Product Name"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiSystemVersion" \
   "Host System Version"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiSystemSerial" \
   "Host System Serial"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiSystemUuid" \
   "Host System UUID"
VBoxManage setextradata "My VM" \
   "VBoxInternal/Devices/pcbios/0/Config/DmiSystemFamily" \
   "Host System Family"
```

Changing this information can be necessary to provide the DMI information of the host to the guest to prevent Windows from asking for a new product key. On Linux hosts the DMI BIOS information can be obtained with

```
dmidecode -t0
```

and the DMI system information can be obtained with

```
dmidecode -t1
```

10 VirtualBox programming interfaces

VirtualBox comes with comprehensive support for third-party developers. The so-called "Main API" of VirtualBox exposes the entire feature set of the virtualization engine. It is completely documented and available to anyone who wishes to control VirtualBox programmatically. We chose the name "Main API" to differentiate it from other parts of the program that may be publicly accessible.

In fact, internally, the front-end programs `VirtualBox`, `VBoxManage` and `VBoxVRDP` (see chapter 7, *Alternative front-ends; remote virtual machines*, page 87) use nothing but this API as well – there are no hidden backdoors into the virtualization engine for our own front-ends. This ensures the entire Main API is both well-documented and well-tested.

For portability and easier maintenance, this Main API is expressed using the Component Object Model (COM), a interprocess mechanism for software components originally introduced by Microsoft for Microsoft Windows. COM has several advantages: it is language-neutral, meaning that even though all of VirtualBox is internally written in C++, programs written in other languages could communicate with it. COM also cleanly separates interface from implementation, so that external programs need not know anything about the messy and complicated details of VirtualBox internals.

On a Windows host, all parts of VirtualBox will use the COM functionality that is native to Windows. On other hosts (including Linux), VirtualBox comes with a built-in implementation of XPCOM, as originally created by the Mozilla project, which we have enhanced to support interprocess communication on a level comparable to Microsoft COM. Internally, VirtualBox has an abstraction layer that allows the same VirtualBox code to work both with native COM as well as our XPCOM implementation.

There are two ways in which the VirtualBox Main API can be used:

- Starting with version 1.6, VirtualBox ships with a **webservice**, which maps all of the COM API to WSDL and SOAP. This allows for controlling VirtualBox remotely and with any programming language with support for webservices – C++, Java, PHP, Perl, C#/.NET/Mono and probably others. For more information, see chapter 10.1, *The VirtualBox webservice: WSDL and SOAP*, page 128 below.

- If you are familiar with COM and the C++ programming language (or with any other programming language that can handle COM/XPCOM objects, such as Java, Visual Basic, C# or JavaScript), then you can use the **COM/XPCOM API** directly. VirtualBox comes with all necessary files and documentation to build fully functional COM applications. For an introduction, please see chapter 10.2, *The VirtualBox COM API*, page 138 below.

If you wonder which way to choose, the brief answer is that the webservice is probably the easiest way to get something running quickly. However, the webservice with all the XML standards involved comes at a price. If – but only if – you are intimately familiar with C++, you may want to go the COM/XPCOM way.

10.1 The VirtualBox webservice: WSDL and SOAP

Starting with version 1.6, VirtualBox comes with a webservice to which any program can connect. The webservice maps the complete Main API, except for a few interfaces that, for technical reasons, cannot be run remotely.

10.1.1 Introduction

Webservices are a particular type of programming interface. Whereas, with "normal" programming, a program calls an application programming interface (API) defined by another program or the operating system and both sides of the interface have to agree on the calling convention and, in most cases, use the same programming language, webservices use Internet standards such as HTTP and XML to communicate.[1]

In order to successfully use a webservice, a number of things are required – primarily, a webservice that is accepting connections, service descriptions, and then a client that connects to that webservice. The connections are governed by the SOAP standard, which describes how messages are to be exchanged between a service and its clients; the service descriptions are governed by WSDL.

In the case of VirtualBox, this translates into the following three components:

1. The VirtualBox webservice (the "server"): this is the `vboxwebsrv` executable shipped with VirtualBox. Once you start this executable (which acts as a HTTP server on a specific TCP/IP port), clients can connect to the webservice and thus control a VirtualBox installation.

2. VirtualBox also comes with WSDL files that describe the services provided by the webservice. You can find these files in the `/sdk/webservice/` subdirectory of your VirtualBox installation. These files are understood by the webservice toolkits that are shipped with most programming languages and enable you to easily access a webservice.

[1] In some ways, webservices promise to deliver the same thing as CORBA and DCOM did years ago. However, while these previous technologies relied on specific binary protocols and thus proved to be difficult to use between diverging platforms, webservices circumvent these incompatibilities by using text-only standards like HTTP and XML. On the downside (and, one could say, typical of things related to XML), a lot of standards are involved before a webservice can be implemented. Many of the standards invented around XML are used one way or another. As a result, webservices are slow and verbose, and the details can be incredibly messy. The relevant standards here are called SOAP and WSDL, where SOAP describes the format of the messages that are exchanged (an XML document wrapped in an HTTP header), and WSDL is an XML format that describes a complete API provided by a webservice. WSDL in turn uses XML Schema to describe types, which is not exactly terse either. However, as you will see from the samples provided in this chapter, the VirtualBox webservice shields you from these details and is easy to use.

3. A client that connects to the webservice in order to control the VirtualBox installation. Unless you play with some of the samples shipped with VirtualBox, this needs to be written by you.

The method for authenticating is configurable via the VirtualBox configuration file. You can change the setting from the command line on the host with:

```
VBoxManage setproperty websrvauthlibrary default|null|<library>
```

The default setting is to use `VRDPAuth` which uses PAM on Linux to authenticate (see chapter 7.4.4, *RDP authentication*, page 93 for more information on the authentication methods). On Linux this usually means that all accesses are denied, due to the access rights of `/etc/shadow`, so authentication only works if this file is accessible by the user account which runs the webservice process. IIn most setups this is only the case for the root user, unless you change your setup.

For testing purposes it's possible to turn off authentication completely (i.e. grant access to all login attempts) with

```
VBoxManage setproperty websrvauthlibrary null
```

This setting should not be used in a production environment.

10.1.2 A hands-on example

Here's a quick-and-dirty example which enumerates the virtual machines registered with a local VirtualBox installation. (It is assumed that you actually have some virtual machines already, otherwise this sample won't do much.) We will explain the details in the following chapters.

10.1.2.1 Prerequisites

This sample assumes that VirtualBox is installed on a Unix-like operating system (such as Linux or Solaris) as the host. You will need:

- A Java Development Kit (JDK). We will use Java as the sample's programming language in this quick-and-dirty sample, but other languages will follow later in this chapter.

- Apache Axis, which is the most widely used toolkit for webservices under Java. If your distribution does not have it installed, you can get a binary from http://www.apache.org. The following examples assume that you have Axis 1.4 installed.

Note: After more testing, we are now recommending to use Sun JAX-WS to write Java clients for the VirtualBox webservice. Unfortunately, we have not had time to update the below instructions for use with JAX-WS. But Sun's Kohsuke Kawaguchi has written an object-oriented wrapper around the VirtualBox webservice using JAX-WS and has published his work on his blog[a]. This wrapper will be included in the next VirtualBox release, and we will update the documentation accordingly.

[a] At http://weblogs.java.net/blog/kohsuke/archive/2008/05/virtualbox_web.html.

10.1.2.2 Step-by-step description

Perform the following steps:

1. Create a working directory somewhere. Under your VirtualBox installation directory, find the `sdk/webservice/samples/java/` directory and copy the file `clienttest.java` to your working directory.

2. Open a terminal in your working directory. Execute the following command:

   ```
   java org.apache.axis.wsdl.WSDL2Java /path/to/vboxwebService.wsdl
   ```

   The vboxwebService.wsdl file should be located in the `sdk/webservice/` directory of your local VirtualBox installation.

   If this fails, your Apache Axis may not be located on your system classpath, and you may have to adjust the CLASSPATH environment variable. Something like this:

   ```
   export CLASSPATH="/path-to-axis-1_4/lib/*":$CLASSPATH
   ```

   Use the directory where the Axis JAR files are located. Mind the quotes so that your shell passes the "*" character to the java executable without expanding. Alternatively, add a corresponding `-classpath` argument to the "java" call above.

   If the command executes successfully, you should see an "org" directory with subdirectories containing Java source files in your working directory. These classes represent the interfaces that the VirtualBox webservice offers, as described by the WSDL file.

   This is the bit that makes using webservices so attractive to client developers: if a language's toolkit understands WSDL, it can generate large amounts of support code automatically. Clients can then easily use this support code and can be done with just a few lines of code.

3. Next, compile the `clienttest.java` source:

   ```
   javac clienttest.java
   ```

This should yield a "clienttest.class" file.

4. Open a second terminal to start the VirtualBox webservice. Change to the directory where the VirtualBox executables are located and type:

   ```
   ./vboxwebsrv
   ```

   The webservice now waits for connections and will run until you press Ctrl+C in this second terminal. (See chapter 10.1.4, *Command line options of vboxwebsrv*, page 132 below for details on how to run the webservice.)

5. Back in the original terminal where you compiled the Java source, run the resulting binary, which will then connect to the webservice:

   ```
   java clienttest
   ```

   The client sample will connect to the webservice (on localhost, but the code could be changed to connect remotely if the webservice was running on a different machine) and make a number of method calls. It will output the version number of your VirtualBox installation and a list of all virtual machines that are currently registered (with a bit of seemingly random data, which will be explained later).

10.1.2.3 A quick look at the code

The Java code is commented. A few items deserve explanation though:

1. The webservice will only work after a client successfully logs on, and the client should log off again. These are the calls to `IWebsessionManager_logon` and `IWebsessionManager_logoff` in the code.

2. Calls to the webservice return objects, but these are stored in strings, and the comments in the code call these "managed object references", and these must be explicitly released by calling `IManagedObjectRef_release`. This requires more explanation.

10.1.3 Fundamental conventions

If you are familiar with other webservices, you may find the VirtualBox webservice to behave a bit differently to accomodate for the fact that VirtualBox webservice more or less maps the VirtualBox Main COM API. The following main differences had to be taken care of:

- Webservices, as expressed by WSDL, are not object-oriented. Even worse, they are normally stateless (or, in webservices terminology, "loosely coupled"). Webservice operations are entirely procedural, and one cannot normally make assumptions of the state of a webservice between function calls. Most importantly, you cannot normally work on objects that are created by one method call.

- The VirtualBox Main API, being expressed in COM, is object-oriented and works entirely on objects, which are grouped into public interfaces, which in turn have attributes and methods associated with them.

For the VirtualBox webservice, this results in three fundamental conventions:

1. All **function names** in the VirtualBox webservice consist of an interface name and a method name, joined together by an underscore. This is because there are only functions ("operations") in WSDL, but no classes, interfaces, or methods.

2. All calls to the VirtualBox webservice (except for logon, see below) take a **managed object reference** as the first argument, representing the object upon which the underlying method is invoked. (Managed object references are explained in detail below.)

   So, one would normally code, in the pseudo-code of an object-oriented language, to invoke a method upon an object:

   ```
   IMachine machine;
   result = machine->getName();
   ```

   In the VirtualBox webservice, this looks something like this (again, pseudo-code):

   ```
   IMachineRef machine;
   result = IMachine_getName(machine);
   ```

3. To make the webservice stateful, and objects persistent between method calls, the VirtualBox webservice introduces a **session manager** (by way of the `IWebsessionManager` interface), which manages object references. Any client wishing to interact with the webservice must first log on to the session manager and in turn receives a managed object reference to an object that supports the `IVirtualBox` interface (the basic interface in the Main API).

In other words, as opposed to other webservices, **the VirtualBox webservice is both object-oriented and stateful.**

10.1.4 Command line options of vboxwebsrv

The machine on which VirtualBox is installed hosts the webservice. This is implemented in a separate executable, `vboxwebsrv`, which, wenn running, allows other programs to connect to it – remotely or from the same machine.

The `vboxwebsrv` program, which implements the webservice, is a text-mode (console) program which, after being started, simply runs until it is interrupted with Ctrl-C or a kill command. It supports the following command line options:

- `-help` (or `-h`): print a brief summary of command line options.

- `-host` (or `-H`): This specifies the host to bind to and defaults to "localhost".

- `-port` (or `-p`): This specifies which port to bind to on the host and defaults to 18083.

- `-timeout` (or `-t`): This specifies the session timeout, in seconds, and defaults to 20. A webservice client that has logged on but makes no calls to the webservice will automatically be disconnected after the number of seconds specified here, as if it had called the `IWebSessionManager::logoff()` method provided by the webservice itself.

  It is normally vital that each webservice client call this method, as the webservice can accumulate large amounts of memory when running, especially if a webservice client does not properly release managed object references. As a result, this timeout value should not be set too high, especially on machines with a high load on the webservice, or the webservice may eventually deny service.

- `-check-interval` (or `-i`): This specifies the interval in which the webservice checks for timed-out clients, in seconds, and defaults to 5. This normally does not need to be changed.

10.1.5 Example: A typical webservice client session

A typical short webservice session to retrieve the version number of the VirtualBox webservice (to be precise, the underlying Main API version number) looks like this:

1. A client logs on to the webservice by calling `IWebsessionManager::logon()` with a valid user name and password. (The webservice can be configured to use various authentication methods, or to let anyone in, which obviously is not optimal for a production environment, as the webservice allows access to the entire VirtualBox API.)

2. On the server side, `vboxwebsrv` creates a session, which persists until the client calls `IWebsessionManager::logoff()` or the session times out after a configurable period of inactivity (see chapter 10.1.4, *Command line options of vboxwebsrv*, page 132).

   For the new session, the webservice creates an instance of `IVirtualBox`. This interface is the most central one in the Main API and allows access to all other interfaces, either through attributes or method calls. For example, IVirtualBox contains a list of all virtual machines that are currently registered (as they would be listed on the left side of the VirtualBox main program).

   The webservice then creates a managed object reference for this instance of `IVirtualBox` and returns it to the calling client, which receives it as the return value of the `IWebsessionManager::logon()` call. Something like this:

   ```
   string oVirtualBox;
   oVirtualBox = webservice->IWebsessionManager_logon("user", "pass");
   ```

(The managed object reference "oVirtualBox" is just a string consisting of digits and dashes. However, it is a string with a meaning and will be checked by the webservice. For details, see below. As hinted above, `IWebsessionManager::logon()` is the *only* operation provided by the webservice which does not take a managed object reference as the first argument!)

3. The VirtualBox Main API documentation says that the `IVirtualBox` interface has a "version" attribute, which is a string. For each attribute, there is a "get" and a "set" method in COM, which maps to according operations in the webservice. So, to retrieve the "version" attribute of this `IVirtualBox` object, the webservice client does this:

```
string version;
version = webservice->IVirtualBox_getVersion(oVirtualBox);

print version;
```

   And it will print "1.6.2".

4. The webservice client calls `IWebsessionManager::logoff()` with the VirtualBox managed object reference. This will clean up all allocated resources.

10.1.6 Managed object references

To a webservice client, a managed object reference looks like a string: two 64-bit hex numbers separated by a dash. This string, however, represents a COM object that "lives" in the webservice process. The two 64-bit numbers encoded in the managed object reference represent a session ID (which is the same for all objects in the same webservice session, i.e. for all objects after one logon) and a unique object ID within that session.

Managed object references are created in two situations:

1. When a client logs on, by calling `IWebsessionManager::logon()`.

   Upon logon, the websession manager creates one instance of `IVirtualBox` and another object of `ISession` representing the webservice session. This can be retrieved using `IWebsessionManager::getSessionObject()`.

   (Technically, there is always only one `IVirtualBox` object, which is shared between all sessions and clients, as it is a COM singleton. However, each session receives its own managed object reference to it. The `ISession` object, however, is created and destroyed for each session.)

2. Whenever a webservice clients invokes an operation whose COM implementation creates COM objects.

   For example, `IVirtualBox::createMachine()` creates a new instance of `IMachine`; the COM object returned by the COM method call is then wrapped into a managed object reference by the webserver, and this reference is returned to the webservice client.

Internally, in the webservice process, each managed object reference is simply a small data structure, containing a COM pointer to the "real" COM object, the session ID and the object ID. This structure is allocated on creation and stored efficiently in hashes, so that the webservice can look up the COM object quickly whenever a webservice client wishes to make a method call. The random session ID also ensures that one webservice client cannot intercept the objects of another.

Managed object references are not destroyed automatically and must be released by explicitly calling `IManagedObjectRef::release()`. This is important, as otherwise hundreds or thousands of managed object references (and corresponding COM objects, which can consume much more memory!) can pile up in the webservice process and eventually cause it to deny service.

To reiterate: The underlying COM object, which the reference points to, is only freed if the managed object reference is released. It is therefore vital that webservice clients properly clean up after the managed object references that are returned to them.

When a webservice client calls `IWebsessionManager::logoff()`, all managed object references created during the session are automatically freed. For short-lived sessions that do not create a lot of objects, logging off may therefore be sufficient, although it is certainly not "best practice".

10.1.7 Some more detail about webservice operation

10.1.7.1 SOAP messages

Whenever a client makes a call to a webservice, this involves a complicated procedure internally. These calls are remote procedure calls. Each such procedure call typically consists of two "message" being passed, where each message is a plain-text HTTP request with a standard HTTP header and a special XML document following. This XML document encodes the name of the procedure to call and the argument names and values passed to it.

To give you an idea of what such a message looks like, assuming that a webservice provides a procedure called "SayHello", which takes a string "name" as an argument and returns "Hello" with a space and that name appended, the request message could look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:test="http://test/">
<SOAP-ENV:Body>
   <test:SayHello>
    <name>Peter</name>
   </test:SayHello>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A similar message – the "response" message – would be sent back from the webservice to the client, containing the return value "Hello Peter".

Most programming languages provide automatic support to generate such messages whenever code in that programming language makes such a request. In other words, these programming languages allow for writing something like this (in pseudo-C++ code):

```
WebServiceClass service("localhost", 18083); // server and port
string result = service.SayHello("Peter");  // invoke remote procedure
```

and would, for these two pseudo-lines, automatically perform these steps:

1. prepare a connection to a webservice running on port 18083 of "localhost";
2. for the `SayHello()` function of the webservice, generate a SOAP message like in the above example by encoding all arguments of the remote procedure call (which could involve all kinds of type conversions and complex marshalling for arrays and structures);
3. connect to the webservice via HTTP and send that message;
4. wait for the webservice to send a response message;
5. decode that response message and put the return value of the remote procedure into the "result" variable.

10.1.7.2 Service descriptions in WSDL

In the above explanations about SOAP, it was left open how the programming language learns about how to translate function calls in its own syntax into proper SOAP messages. In other words, the programming language needs to know what operations the webservice supports and what types of arguments are required for the operation's data in order to be able to properly serialize and deserialize the data to and from the webservice. For example, if a webservice operation expects a number in "double" floating point format for a particular parameter, the programming language cannot send to it a string instead.

For this, the Web Service Definition Language (WSDL) was invented, another XML substandard that describes exactly what operations the webservice supports and, for each operation, which parameters and types are needed with each request and reponse message. WSDL descriptions can be incredibly verbose, and one of the few good things that can be said about this standard is that it is indeed supported by most programming languages.

So, if it is said that a programming language "supports" webservices, this typically means that a programming language has support for parsing WSDL files and somehow integrating the remote procedure calls into the native language syntax – for example, like in the Java sample shown in chapter 10.1.2, *A hands-on example*, page 129.

For details about how programming languages support webservices, please refer to the documentation that comes with the individual languages. Here are a few pointers:

1. For **C++,** among many others, the gSOAP toolkit is a good option. Parts of gSOAP are also used in VirtualBox to implement the VirtualBox webservice.

2. For **Java,** there are several implementations – one of them being Axis by the Apache foundation, as described in chapter 10.1.2, *A hands-on example*, page 129.

3. **Perl** supports WSDL via the SOAP::Lite package. This in turn comes with a tool called `stubmaker.pl` that allows you to turn any WSDL file into a Perl package that you can import. (You can also import any WSDL file "live" by having it parsed every time the script runs, but that can take a while.) You can then code (again, assuming the above example):

   ```
   my $result = servicename->sayHello("Peter");
   ```

   A sample webservice client written in Perl is also shipped with VirtualBox.

10.1.8 Using the VirtualBox Main API documentation for webservice clients

The VirtualBox COM API is broken into many interfaces, which map to classes in C++. The complete API is documented in `VirtualBoxAPI.chm`, a help file in Windows Help format that can be viewed on both Windows and Linux (in the latter case, for example, with the `kchmviewer` program that ships with VirtualBox).

This file documents all interfaces, attributes and methods provided by the Main API. However, as indicated above, "interface", "attribute" and "method" are COM concepts, so the API documentation needs to be read with the following in mind:

1. Any method call becomes a **function call** in the webservice, with the object as the first parameter. So when the documentation says that the `IVirtualBox` interface supports the `createMachine()` method, the webservice operation is `IVirtualBox_createMachine()`, and a managed object reference to an `IVirtualBox` object must be passed as the first argument.

2. For **attributes** in interfaces, there will be at least one "get" function; there will also be a "set" function, unless the attribute is "readonly". The attribute name will be appended to the "get" or "set" prefix, with a capizalized first letter. So, the "version" readonly attribute of the `IVirtualBox` interface can be retrieved by calling `IVirtualBox_getVersion()`.

3. Whenever the API documentation says that a method (or an attribute getter) returns an **object**, it will returned a managed object reference in the webservice instead. As said above, managed object references should be released if the webservice client does not log off again immediately!

4. COM does not support **arrays**. As a result, the Main API needs to work around this limitation by using **"collections"** of objects wherever a list of things might be

returned. For example, `IVirtualBox` has an attribute "machines", which is of the `IMachineCollection` type. However, arrays are supported in SOAP and WSDL, so whenever you see a "Collection" of something in the API documentation, you can be sure that it is an array in your client programming language.

For example, the "machines" attribute of `IVirtualBox`, of which the API documentation says it is an `IMachineCollection`, will be an array of `IMachine` managed object references in your programming language.

Note: When an array of managed object references is returned by a function, you must release each managed object reference!

As was indicated above, the most central interface in the VirtualBox Main API is `IVirtualBox`. Virtual machines, which most programmers will also be interested in, are represented by the `IMachine` interface. Also, in order to do anything interesting with a virtual machine (such as changing its settings or starting it), one needs to create a session object first; this is of the `ISession` interface. It is recommended to read through the documentation of these three interfaces first to get a basic grip on the Main API.

10.2 The VirtualBox COM API

If you do not require remote procedure calls such as those offered by the VirtualBox webservice, and if you know C++ and COM, you may elect to program VirtualBox's Main API directly via COM.

VirtualBox ships with a sample programs that demonstrate how to use the Main API to implement a number of tasks on your host platform. These samples can be found in the `/sdk/samples/API` directory below the VirtualBox installation directory and will be called `tstVBoxAPIWin.cpp` on Windows or `txtVBoxAPILinux.cpp` on the other host platforms. The two samples are actually different, because the one for Windows uses native COM, whereas the other uses our XPCOM implementation, as described above.

Since COM and XPCOM are conceptually very similar but vary in the implementation details, we have created a "glue" layer that shields COM client code from these differences. All VirtualBox uses only this glue layer, so the same code written once works on both Windows hosts (with native COM) as well as on other hosts (with our XPCOM implementation). It is recommended to always use this glue code instead of using the COM and XPCOM APIs directly, as it is very easy to make your code completely independent from the platform it is running on. A third sample, `tstVBoxAPIGlue.cpp`, illustrates how to use the glue layer.

In order to encapsulate platform differences between Microsoft COM and XPCOM, the following items should be kept in mind when using the glue layer:

1. **Attribute getters and setters.** COM has the notion of "attributes" in interfaces, which roughly compare to C++ member variables in classes. The difference is that for each attribute declared in an interface, COM automatically provides a "get" method to return the attribute's value. Unless the attribute has been marked as "readonly", a "set" attribute is also provided.

   To illustrate, the IVirtualBox interface has a "version" attribute, which is read-only and of the "wstring" type (the standard string type in COM). As a result, you can call the "get" method for this attribute to retrieve the version number of VirtualBox.

   Unfortunately, the implementation differs between COM and XPCOM. Microsoft COM names the "get" method like this: `get_Attribute()`, whereas XPCOM uses this syntax: `GetAttribute()` (and accordingly for "set" methods). To hide these differences, the VirtualBox glue code provides the `COMGETTER(attrib)` and `COMSETTER(attrib)` macros. So, `COMGETTER(version)()` (note, two pairs of brackets) expands to `get_Version()` on Windows and `GetVersion()` on other platforms.

2. **Unicode conversions.** While the rest of the modern world has pretty much settled on encoding strings in UTF-8, COM, unfortunately, uses UCS-16 encoding. This requires a lot of conversions, in particular between the VirtualBox Main API and the Qt GUI, which, like the rest of Qt, likes to use UTF-8.

   To facilitate these conversions, VirtualBox provides the `com::Bstr` and `com::Utf8Str` classes, which support all kinds of conversions back and forth.

3. **COM autopointers.** Possibly the greatest pain of using COM – reference counting – is alleviated by the `ComPtr<>` template provided by the `ptr.h` file in the glue layer.

11 Troubleshooting

This chapter provides answers to commonly asked questions. In order to improve your user experience with VirtualBox, it is recommended to read this section to learn more about common pitfalls and get recommendations on how to use the product.

11.1 General

11.1.1 Collecting debugging information

For problem determination, it is often important to collect debugging information which can be analyzed by VirtualBox support. This section contains information about what kind of information can be obtained.

Every time VirtualBox starts up a VM, a log file is created containing some information about the VM configuration and runtime events. The log file is called `VBox.log` and resides in the VM log file folder. Typically this will be a directory like this:

```
$HOME/.VirtualBox/Machines/{machinename}/Logs
```

When starting a VM, the configuration file of the last run will be renamed to `.1`, up to `.3`. Sometimes when there is a problem, it is useful to have a look at the logs. Also when requesting support for VirtualBox, supplying the corresponding log file is mandatory.

For convenience, for each virtual machine, the VirtualBox main window can show these logs in a window. To access it, select a virtual machine from the list on the left and select “Show logs...“ from the “Machine” window.

11.1.2 Guest shows IDE errors for VDI on slow host file system

Occasionally, some host file systems provide very poor writing performance and as a consequence cause the guest to time out IDE commands. This is normal behavior and should normally cause no real problems, as the guest should repeat commands that have timed out. However some guests (e.g. some Linux versions) have severe problems if a write to a VDI file takes longer than about 15 seconds. Some file systems however require more than a minute to complete a single write, if the host cache contains a large amount of data that needs to be written.

The symptom for this problem is that the guest can no longer access its files during large write or copying operations, usually leading to an immediate hang of the guest.

In order to work around this problem (the true fix is to use a faster file system that doesn’t exhibit such unacceptable write performance), it is possible to flush the VDI

after a certain amount of data has been written. This interval is normally infinite, but can be configured individually for each disk of a VM using the following command:

```
VBoxManage setextradata <vmname>
           "VBoxInternal/Devices/piix3ide/0/LUN#[x]/Config/FlushInterval" [b]
```

The value [x] that selects the disk is 0 for the master device on the first channel, 1 for the slave device on the first channel, 2 for the master device on the second channel or 3 for the master device on the second channel. Only disks support this configuration option. It must not be set for CD-ROM drives.

The unit of the interval [b] is the number of bytes written since the last flush. The value for it must be selected so that the occasional long write delays do not occur. Since the proper flush interval depends on the performance of the host and the host filesystem, finding the optimal value that makes the problem disappear requires some experimentation. Values between 1000000 and 10000000 (1 to 10 megabytes) are a good starting point. Decreasing the interval both decreases the probability of the problem and the write performance of the guest. Setting the value unnecessarily low will cost performance without providing any benefits. An interval of 1 will cause a flush for each write operation and should solve the problem in any case, but has a severe write performance penalty.

Providing a value of 0 for [b] is treated as an infinite flush interval, effectively disabling this workaround. Removing the extra data key by specifying no value for [b] has the same effect.

11.1.3 Responding to guest IDE flush requests

If desired, the virtual disk images (VDI) can be flushed when the guest issues the IDE FLUSH CACHE command. Normally these requests are ignored for improved performance. To enable flushing, issue the following command:

```
VBoxManage setextradata <vmname>
           "VBoxInternal/Devices/piix3ide/0/LUN#[x]/Config/IgnoreFlush" 0
```

The value [x] that selects the disk is 0 for the master device on the first channel, 1 for the slave device on the first channel, 2 for the master device on the second channel or 3 for the master device on the second channel. Only disks support this configuration option. It must not be set for CD-ROM drives.

Note that this doesn't affect the flushes performed according to the configuration described in chapter 11.1.2, *Guest shows IDE errors for VDI on slow host file system*, page 140. Restoring the default of ignoring flush commands is possible by setting the value to 1 or by removing the key.

11.2 Windows guests

11.2.1 Windows boot failures (bluescreens) after changing VM configuration

Often, customers encounter Windows startup failures (the infamous “blue screen”) after performing configuration changes to a virtual machine which are not allowed for an already installed Windows operating system. Depending on the presence of several hardware features, the Windows installation program chooses special kernel and device driver versions and will fail to startup should these hardware features be removed.

Most importantly, never disable ACPI and the I/O APIC if they were enabled at installation time. Enabling them for a Windows VM which was installed without them does not cause any harm. However, Windows will not use these features in this case.

11.2.2 Windows 2000 installation failures

When installing Windows 2000 guests, you might run into one of the following issues:

- Installation reboots, usually during component registration.
- Installation fills the whole hard disk with empty log files.
- Installation complains about a failure installing msgina.dll.

These problems are all caused by a bug in the hard disk driver of Windows 2000. After issuing a hard disk request, there is a race condition in the Windows driver code which leads to corruption if the operation completes too fast, i.e. the hardware interrupt from the IDE controller arrives too soon. With physical hardware, there is a guaranteed delay in most systems so the problem is usually hidden there (however it should be possible to reproduce it on physical hardware as well). In a virtual environment, it is possible for the operation to be done immediately (especially on very fast systems with multiple CPUs) and the interrupt is signaled sooner than on a physical system. The solution is to introduce an artificial delay before delivering such interrupts. This delay can be configured for a VM using the following command:

```
VBoxManage setextradata <vmname>
           "VBoxInternal/Devices/piix3ide/0/Config/IRQDelay" 1
```

This sets the delay to one millisecond. In case this doesn’t help, increase it to a value between 1 and 5 milliseconds. Please note that this slows down disk performance. After installation, you should be able to remove the key (or set it to 0).

11.2.3 How to record bluescreen information from Windows guests

When Windows guests run into a kernel crash, they display the infamous bluescreen. Depending on how Windows is configured, the information will remain on the screen

until the machine is restarted or it will reboot automatically. During installation, Windows is usually configured to reboot automatically. With automatic reboots, there is no chance to record the bluescreen information which might be important for problem determination.

VirtualBox provides a method of halting a guest when it wants to perform a reset. In order to enable this feature, issue the following command:

```
VBoxManage setextradata <vmname>
           "VBoxInternal/PDM/HaltOnReset" 1
```

11.2.4 No networking in Windows Vista guests

Unfortunately, with Vista, Microsoft dropped support for the virtual AMD PCnet card that we are providing to virtual machines. As a result, after installation, Vista guests initially have no networking. VirtualBox therefore ships a driver for that card with the Windows Guest Additions; see chapter 4.2.4, *Windows Vista networking*, page 55.

11.2.5 Windows guests may cause a high CPU load

Several background applications of Windows guests, especially virus scanners, are known to increases the CPU load notably even if the guest appears to be idle. We recommend to deactivate virus scanners within virtualized guests if possible.

11.3 Linux guests

11.3.1 Linux guests may cause a high CPU load

Some Linux guests may cause a high CPU load even if the guest system appears to be idle. This can be caused by a high timer frequency of the guest kernel. Some Linux distributions, for example Fedora, ship a Linux kernel configured for a timer frequency of **1000Hz**. We recommend to recompile the guest kernel and to select a timer frequency of 100Hz.

11.4 Windows hosts

11.4.1 VBoxSVC out-of-process COM server issues

VirtualBox makes use of the Microsoft Component Object Model (COM) for inter- and intra-process communication. This allows VirtualBox to share a common configuration among different virtual machine processes and provide several user interface options based on a common architecture. All global status information and configuration is maintained by the process `VBoxSVC.exe`, which is an out-of-process COM server. Whenever a VirtualBox process is started, it requests access to the COM server and

Windows automatically starts the process. Note that it should never be started by the end user.

When the last process disconnects from the COM server, it will terminate itself after some seconds. The VirtualBox configuration (XML files) is maintained and owned by the COM server and the files are locked whenever the server runs.

In some cases - such as when a virtual machine is terminated unexpectedly - the COM server will not notice that the client is disconnected and stay active for a longer period (10 minutes or so) keeping the configuration files locked. In other rare cases the COM server might experience an internal error and subsequently other processes fail to initialize it. In these situations, it is recommended to use the Windows task manager to kill the process `VBoxSVC.exe`.

11.4.2 CD/DVD changes not recognized

In case you have assigned a physical CD/DVD drive to a guest and the guest does not notice when the medium changes, make sure that the Windows media change notification (MCN) feature is not turned off. This is represented by the following key in the Windows registry:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Cdrom\Autorun
```

Certain applications may disable this key against Microsoft's advice. If it is set to 0, change it to 1 and reboot your system. VirtualBox relies on Windows notifying it of media changes.

11.4.3 Sluggish response when using Microsoft RDP client

If connecting to a Virtual Machine via the Microsoft RDP client (called Remote Desktop Connection), there can be large delays between input (moving the mouse over a menu is the most obvious situation) and output. This is because this RDP client collects input for a certain time before sending it to the VRDP server built into VirtualBox.

The interval can be decreased by setting a Windows registry key to smaller values than the default of 100. The key does not exist initially and must be of type DWORD. The unit for its values is milliseconds. Values around 20 are suitable for low-bandwidth connections between the RDP client and server. Values around 4 can be used for a gigabit Ethernet connection. Generally values below 10 achieve a performance that is very close to that of the local input devices and screen of the host on which the Virtual Machine is running.

Depending whether the setting should be changed for an individual user or for the system, either

```
HKEY_CURRENT_USER\Software\Microsoft\Terminal Server
        Client\Min Send Interval
```

or

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Terminal Server
        Client\Min Send Interval
```

can be set appropriately.

11.4.4 Running an iSCSI initiator and target on a single system

Deadlocks can occur on a Windows host when attempting to access an iSCSI target running in a guest virtual machine with an iSCSI initiator (e.g. Microsoft iSCSI Initiator) that is running on the host. This is caused by a flaw in the Windows cache manager component, and causes sluggish host system response for several minutes, followed by a "Delayed Write Failed" error message in the system tray or in a separate message window. The guest is blocked during that period and may show error messages or become unstable.

Setting the environment variable `VBOX_DISABLE_HOST_DISK_CACHE` to 1 will enable a workaround for this problem until Microsoft addresses the issue. For example, open a command prompt window and start VirtualBox like this:

```
set VBOX_DISABLE_HOST_DISK_CACHE=1
VirtualBox
```

While this will decrease guest disk performance (especially writes), it does not affect the performance of other applications running on the host.

11.5 Linux hosts

11.5.1 Linux kernel module refuses to load

If the VirtualBox kernel module (`vboxdrv`) refuses to load, i.e. you get an "Error inserting vboxdrv: Invalid argument", check (as root) the output of the `dmesg` command to find out why the load failed. The most common reasons are:

- With Linux 2.6.19 and higher, the NMI watchdog may be active. Add `nmi_watchdog=0` to the kernel command line (e.g. in your grub configuration) and reboot. With the Debian and Ubuntu installation modules, execute `sudo dpkg-reconfigure virtualbox` again.
- The kernel disagrees about the version of the gcc used to compile the module. Make sure that you use the same compiler as used to build the kernel.

11.5.2 Linux host CD/DVD drive not found

If you have configured a virtual machine to use the host's CD/DVD drive, but this does not appear to work, make sure that the current user has permission to access the corresponding Linux device file (`/dev/hdc` or `/dev/scd0` or `/dev/cdrom` or similar). On most distributions, the user must be added to a corresponding group (usually called `cdrom` or `cdrw`).

11.5.3 Linux host CD/DVD drive not found (older distributions)

On older Linux distributions, if your CD/DVD device has a different name, VirtualBox may be unable to find it. On older Linux hosts, VirtualBox performs the following steps to locate your CD/DVD drives:

1. VirtualBox examines if the environment variable `VBOX_CDROM` is defined (see below). If so, VirtualBox omits all the following checks.
2. VirtualBox tests if `/dev/cdrom` works.
3. In addition, VirtualBox checks if any CD/DVD drives are currently mounted by checking `/etc/mtab`.
4. In addition, VirtualBox checks if any of the entries in `/etc/fstab` point to CD/DVD devices.

In other words, you can try to set VBOX_CDROM to contain a list of your CD/DVD devices, separated by colons, for example as follows:

```
export VBOX_CDROM='/dev/cdrom0:/dev/cdrom1'
```

On modern Linux distributions, VirtualBox uses the hardware abstraction layer (hal) to locate CD and DVD hardware.

11.5.4 Linux host floppy not found

The previous instructions (for CD and DVD drives) apply accordingly to floppy disks, except that on older distributions VirtualBox tests for `/dev/fd*` devices by default, and this can be overridden with the `VBOX_FLOPPY` environment variable.

11.5.5 Strange guest IDE error messages when writing to CD/DVD

If the experimental CD/DVD writer support is enabled with an incorrect VirtualBox, host or guest configuration, it is possible that any attempt to access the CD/DVD writer fails and simply results in guest kernel error messages (for Linux guests) or application error messages (for Windows guests). VirtualBox performs the usual consistency checks when a VM is powered up (in particular it aborts with an error message if the device for the CD/DVD writer is not writable by the user starting the VM), but it cannot detect all misconfigurations. The necessary host and guest OS configuration is not specific for VirtualBox, but a few frequent problems are listed here which occurred in connection with VirtualBox.

Special care must be taken to use the correct device. The configured host CD/DVD device file name (in most cases /dev/cdrom) must point to the device that allows writing to the CD/DVD unit. For CD/DVD writer units connected to a SCSI controller or to a IDE controller that interfaces to the Linux SCSI subsystem (common for some SATA controllers), this must refer to the SCSI device node (e.g. /dev/scd0). Even

for IDE CD/DVD writer units this must refer to the appropriate SCSI CD-ROM device node (e.g. /dev/scd0) if the ide-scsi kernel module is loaded. This module is required for CD/DVD writer support with all Linux 2.4 kernels and some early 2.6 kernels. Many Linux distributions load this module whenever a CD/DVD writer is detected in the system, even if the kernel would support CD/DVD writers without the module. VirtualBox supports the use of IDE device files (e.g. /dev/hdc), provided the kernel supports this and the ide-scsi module is not loaded.

Similar rules (except that within the guest the CD/DVD writer is always an IDE device) apply to the guest configuration. Since this setup is very common, it is likely that the default configuration of the guest works as expected.

11.5.6 VBoxSVC IPC issues

On Linux, VirtualBox makes use of a custom version of Mozilla XPCOM (cross platform component object model) for inter- and intra-process communication (IPC). The process `VBoxSVC` serves as a communication hub between different VirtualBox processes and maintains the global configuration, i.e. the XML database. When starting a VirtualBox component, the processes `VBoxSVC` and `VirtualBoxXPCOMIPCD` are started automatically. They are only accessible from the user account they are running under. `VBoxSVC` owns the VirtualBox configuration database which normally resides in `~/.VirtualBox`. While it is running, the configuration files are locked. Communication between the various VirtualBox components and `VBoxSVC` is performed through a local domain socket residing in `/tmp/.vbox-<username>-ipc`. In case there are communication problems (i.e. a VirtualBox application cannot communicate with `VBoxSVC`), terminate the daemons and remove the local domain socket directory.

11.5.7 USB not working

If USB is not working on your Linux host, make sure that the current user has permission to access the USB filesystem (`usbfs`), which VirtualBox relies on to retrieve valid information about your host's USB devices.

As `usbfs` is a virtual filesystem, a `chmod` on `/proc/bus/usb` has no effect. The permissions for `usbfs` can therefore *only* be changed by editing the `/etc/fstab` file.

For example, most Linux distributions have a user group called `usb` or similar, of which the current user must be a member. To give all users of that group access to usbfs, make sure the following line is present:

```
# 85 is the USB group
none     /proc/bus/usb     usbfs      devgid=85,devmode=664    0    0
```

Replace 85 with the group ID that matches your system (search `/etc/group` for "usb" or similar). Alternatively, if you don't mind the security hole, give all users access to USB by changing "664" to "666".

The various distributions are very creative from which script the `usbfs` filesystem is mounted. Sometimes the command is hidden in unexpected places.

For SuSE 10.0 the mount command is part of the udev configuration file `/etc/udev/rules.d/50-udev.rules`. As this distribution has no user group called `usb`, you may e.g. use the `vboxusers` group which was created by the VirtualBox installer. Since group numbers are allocated dynamically, the following example uses 85 as a placeholder. Modify the line containing (a linebreak has been inserted to improve readability)

```
DEVPATH="/module/usbcore", ACTION=="add",
    RUN+="/bin/mount -t usbfs usbfs /proc/bus/usb"
```

and add the necessary options (make sure that everything is in a single line):

```
DEVPATH="/module/usbcore", ACTION=="add",
    RUN+="/bin/mount -t usbfs usbfs /proc/bus/usb -o devgid=85,devmode=664"
```

Debian Etch has the mount command in `/etc/init.d/mountkernfs.sh`. Since that distribution has no group `usb`, it is also the easiest solution to allow all members of the group `vboxusers` to access the USB subsystem. Modify the line

```
domount usbfs usbdevfs /proc/bus/usb -onoexec,nosuid,nodev
```

so that it contains

```
domount usbfs usbdevfs /proc/bus/usb -onoexec,nosuid,nodev,devgid=85,devmode=664
```

As usual, replace the 85 with the actual group number which should get access to USB devices.

Other distributions do similar operations in scripts stored in the `/etc/init.d` directory.

11.5.8 PAX/grsec kernels

Linux kernels including the grsec patch (see http://www.grsecurity.net/) and derivates have to disable PAX_MPROTECT for the VBox binaries to be able to start a VM. The reason is that VBox has to create executable code on anonymous memory.

11.5.9 Linux kernel vmalloc pool exhausted

When running a large number of VMs with a lot of RAM on a Linux system (say 20 VMs with 1GB of RAM each), additional VMs might fail to start with a kernel error saying that the vmalloc pool is exhausted and should be extended. The error message also tells you to specify `vmalloc=256MB` in your kernel parameter list. If adding this parameter to your GRUB or LILO configuration makes the kernel fail to boot (with a weird error message such as failed to mount the root partition), then you probably run into a memory conflict of your kernel and initial RAM disk. This can be solved by adding the following parameter to your GRUB configuration:

```
uppermem 524288
```

.

12 Change log

This section summarizes the changes between VirtualBox versions. Note that this change log is not exhaustive; not all changes are listed.

VirtualBox version numbers consist of three numbers separated by dots where the first number represents the major version, the 2nd number the minor version and the 3rd one the build number. Build numbers of official releases are always even. An odd build number represents an internal development or test build.

12.1 Version 1.6.2 (2008-05-28)

This is a maintenance release. The following items were fixed and/or added:

- GUI: fixed a bug which prevented to add more than one SATA drive from the GUI
- GUI: fixed a regression introduced in 1.6.0: the fullscreen mode was left on every guest video mode switch
- GUI: fixed several minor issues
- Networking: fixed a host interface networking regression introduced in 1.6.0
- VMM: fixed starting of VMs with AMD-V enabled
- VMM: massive performance enhancements for AMD-V
- VMM: stability improvements for AMD-V on Windows hosts
- VMM: correctly detect AMD CPUs with erratum 170 (AMD-V)
- VMM: detect inconsistent timestamp counters on certain AMD Phenom CPUs (Windows host only)
- VMM: fixed KVM check (Linux hosts only)
- VMM: fixed a regression introduced in 1.6.0: Windows stuck during installation
- XPCOM: fixed several races
- SATA: improved performance with Vista guests
- SATA: fixed statistics counter
- Shared Folders: several fixes (iTunes download, speed up browsing)

- ATA/IDE: fixed boot from CDROM if a medium was added while the boot menu was active
- Networking: provide an *Intel PRO/1000 T Server (82543GC)* network device emulation which is recognized by Windows XP guests
- Networking: fixes for the E1000 emulation (don't crash if not attached, fixed a bug in the statistics counter implementation)
- NAT: don't crash if the guest sent a DHCPRELEASE message with an invalid IP address
- NAT: fixed ARP reply for the NAT gateway and for the NAT name server if the guest IP range was changed
- Internal Networking: fixed shutdown if more than two VMs are connected to the same network
- BIOS: allow to change the DMI informatiton (see chapter 9.13, *Configuring the BIOS DMI information*, page 125)
- RTC: fixed UIP emulation to prevent jumping of time in Solaris guests
- Windows host: VirtualBox installation directory corrected for 64 bits Windows
- Windows host: fixed VBoxVRDP.exe symlink
- Windows host: solved locking problems in raw partition VMDK support
- Windows host: fixed stability during high system load (page fault in KeQueryActiveProcessors)
- MacOS X host: fixed crashes under certain conditions
- Shared Folders: limited users without admin rights now also can use Shared Folders on Windows guests
- Linux hosts: fixed default runlevel for the kernel module helper script
- Solaris hosts: enabled support for VT-x and AMD-V
- Solaris hosts: dynamic loading of libdlpi fixes a problem where Solaris 10 was not able to start a VM
- Linux additions: fixed runlevels for kernel module helper scripts
- Linux additions: compatibility fixes with Linux 2.6.26
- Linux additions: fixed occasional guest kernel crash during unload of the vboxadd guest kernel module
- X11 Guest Additions: fixed a problem preventing clipboard transfers over 1K from host to guest

12.2 Version 1.6.0 (2008-04-30)

This version is a major update. The following major new features were added:

- Solaris and Mac OS X host support
- Seamless windowing for Linux and Solaris guests
- Guest Additions for Solaris
- A webservice API (see chapter 10, *VirtualBox programming interfaces*, page 127)
- SATA hard disk (AHCI) controller (see chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62)
- Experimental Physical Address Extension (PAE) support

In addition, the following items were fixed and/or added:

- GUI: added accessibility support (508)
- GUI: VM session information dialog
- VBoxHeadless: renamed from VBoxVRDP
- VMM: reduced host CPU load of idle guests
- VMM: many fixes for VT-x/SVM hardware-supported virtualization
- ATA/IDE: better disk geometry compatibility with VMware images
- ATA/IDE: virtualize an AHCI controller
- Storage: better write optimization, prevent images from growing unnecessarily.
- Network: support PXE booting with NAT
- Network: fixed the Am79C973 PCNet emulation for Nexenta guests
- NAT: improved builtin DHCP server (implemented DHCPNAK response)
- NAT: port forwarding stopped when restoring the VM from a saved state
- NAT: make subnet configurable
- XPCOM: moved to libxml2
- XPCOM: fixed VBoxSVC autostart race
- Audio: SoundBlaster 16 emulation
- USB: fixed problems with USB 2.0 devices

- MacOS X: fixed seamless mode
- MacOS X: better desktop integration, several look'n'feel fixes
- MacOS X: switched to Quartz2D framebuffer
- MacOS X: added support for shared folders
- MacOS X: added support for clipboard integration
- Solaris: added host audio playback support (experimental)
- Solaris: made it possible to run VirtualBox from non-global zones
- Shared Folders: many bugfixes to improve stability
- Seamless windows: added support for Linux guests
- Linux installer: support DKMS for compiling the kernel module
- Linux host: compatibility fixes with Linux 2.6.25
- Windows host: support for USB devices has been significantly improved; many additional USB devices now work
- Windows Additions: automatically install AMD PCNet drivers on Vista guests
- Linux additions: several fixes, experimental support for RandR 1.2
- Linux additions: compatibility fixes with Linux 2.6.25

12.3 Version 1.5.6 (2008-02-19)

This version is a maintenance release. It adds an experimental Intel Gigabit Ethernet device emulation and read-only shared folders.

- GUI: fixed several error messages
- GUI: fixed registration dialog crashes once and for all
- GUI: really ask before resetting the VM
- GUI: release mouse and keyboard before the host activates the screensaver
- GUI: fixed issue with license display on big screens
- GUI: added setting for network name for internal networks
- GUI: added setting for network device type
- GUI: keyboard fixes

- GUI: seamless mode and fullscreen mode fixes
- GUI: fixed soaked hostkey keyup event under certain conditions
- GUI: more informative message dialog buttons
- GUI: VM selector context menu
- VBoxSDL: added -termacpi switch
- VBoxSDL: fixed automatic adaption of the guest screen resolution to the size of the VM window
- VMM: under heavy guest activity, for example when copying files to/from a shared folder, the VM could crash with an assertion
- VMM: added an option to select PIIX4 (improves compatibility with Windows guests created by VMware)
- VMM: fixed a bug which could lead to memory corruption under rare circumstances
- VMM: improved performance of Solaris guests
- VRDP: fixed a 1.5.4 regression: VRDP client and server were out-of-sync if the VM was started using the GUI
- VRDP: proper error handling if the VRDP library could not be loaded
- VRDP: fixed compilation of the Linux rdesktop client on newer Linux kernels
- VRDP: install rdesktop-vrdp on Linux hosts
- VBoxManage: fixed crash during clonevdi
- VBoxManage: added 'list runningvms' command
- VBoxManage: improved the compatibility when reading the partition table of a raw disk
- Shared Folders: added support for read-only shared folders
- Shared Clipboard: several fixes
- Network: don't crash if the device is activated but not attached
- Network: experimental support for Intel Gigabit Ethernet (E1000) device emulation
- iSCSI: better check for misconfigured targets
- iSCSI: allow to directly attach to internal networks with integrated mini IP stack

- PulseAudio: don't hang during VM initialization if no sound server is available
- VDI: fixed sized virtual disk images are now completely written during creation to workaround buggy sparse file handling on some OS (e.g. Vista)
- VDI/VMDK: prevent indexing of .vdi and .vmdk files on Windows hosts
- ACPI: added sleep button event
- Serial: proper handling of inaccessible host devices
- Windows installer: allow smooth upgrade without deinstallation
- Linux installer: fixed Slackware detection regression
- Linux installer: updated VBoxTunctl allowing to assign a tap device to a group on Linux kernels > 2.6.23
- Windows additions: several fixes, in particular for Windows NT4
- Windows additions: made them uninstallable
- Linux additions: fixed installer for Kubuntu 8.04
- Linux additions: add default video mode for handling video mode hints from the host
- Linux host: compatibility fixes with Linux > 2.6.24

12.4 Version 1.5.4 (2007-12-29)

This version is a maintenance release. It adds USB 2.0 support and a PulseAudio backend.

- GUI: fixed registration dialog crashes
- GUI: allow to enter unicode characters to the name of the registration dialog
- GUI: pre-select attached media in the disk manager when opened from the VM settings dialog
- GUI: remember the last active VM
- GUI: ask before reset the VM
- GUI: don't accept empty paths for serial/parallel ports in XML
- GUI: fixed NumLock / CapsLock synchronizazion on Windows hosts
- GUI: don't start the kernel timer if no VM is active (Linux host)

- GUI: fixed accelerators in German translation
- VMM: improved compatibility with Solaris guests
- VMM: properly restore CR4 after leaving VT-x mode
- VMM: fix interrupt storm with Windows guests under certain circumstances (e.g. disable + re-enable the network adapter)
- VMM: with VT-x a pending interrupt could be cleared behind our back
- VMM: workaround for missed cpuid patch (some Linux guests refuse to boot on multi-core CPUs)
- VMM: fixed code for overriding CPUID values
- VMM: improved error handling on out-of-memory conditions
- API: don't crash when trying to create a VM with a duplicate name
- API: don't crash when trying to access the settings of a VM when some other VMs are not accessible
- API: fixed several memory leaks
- ATA/IDE: fixed SuSE 9.1 CD read installer regression
- Serial: several fixes
- Floppy: fixed inverted write protect flag
- Floppy: fixed handling of read-only images
- USB: virtualize an EHCI controller
- USB: several minor fixes
- Network: fixed MAC address check
- Network: host interface fixes for Solaris guests
- Network: guest networking stopped completely after taking a snapshot
- Network: don't crash if a network card is enabled but not attached
- PXE: fix for PXE-EC8 error on soft reboot
- NAT: update the DNS server IP address on every DNS packet sent by the guest
- VGA: reset VRAM access handers after a fullscreen update
- VGA: don't overwrite guest's VRAM when displaying a blank screen
- ACPI: implemented the sleep button event

- VRDP: fixed crash when querying VRDP properties
- VRDP: netAddress fixes
- VRDP: fixed the Pause/Break keys over VRDP
- VRDP: workaround for scrambled icons with a guest video mode of 16bpp
- VRDP: reset modifer keys on RDP_INPUT_SYNCHRONIZE
- VRDP: reset RDP updates after resize to prevent obsolete updates
- Clipboard: Windows host/guest fixes
- Clipboard: fixed a SEGFAULT on VM exit (Linux host)
- Clipboard: fixed a buffer overflow (Linux host)
- Shared Folders: fixed memory leaks
- Linux installer: remove the old kernel module before compiling a new one
- Linux host: compatibility fixes with Linux 2.6.24
- Linux host: script fixes for ArchLinux
- Linux host: load correct HAL library to determine DVD/floppy (libhal.so.1 not libhal.so)
- Linux host: make sure the tun kernel module is loaded before initializing static TAP interfaces
- Windows additions: fixed hang during HGCM communication
- Windows additions: fixed delay when shutting down the guest
- Linux additions: added sendfile support to allow HTTP servers to send files on shared folders
- Linux additions: make additions work with Fedora 8 (SELinux policy added)
- Linux additions: sometimes ARGB pointers were displayed incorrectly
- Linux additions: several small script fixes

12.5 Version 1.5.2 (2007-10-18)

This version is a maintenance release and mainly addresses issues discovered in VirtualBox 1.5.0 and improves compatibility with new guest and host OS revisions

- Windows Installer: fixed installation on Windows 2000 hosts
- Windows Installer: proper warning when installing a 32-bit VirtualBox version on 64-bit Windows and vice versa
- Linux Installer: no longer require license acceptance during install, instead at first GUI startup (addresses issues with hanging installer on Debian based distributions)
- GUI: added user registration dialog
- GUI: fixed crashes on 64-bit Linux hosts
- GUI: several fixes and improvements to seamless mode
- GUI: fixed DirectDraw mode with certain video cards (e.g. Intel i915)
- GUI: fixed incorrect guest resolution after leaving fullscreen mode
- GUI: improved keyboard handling on Linux host
- GUI: show fatal VM aborts (aka “Guru Meditation”)
- GUI: fixed crashes due to a display update race condition on some systems
- GUI: added ACPI shutdown option to the VM close dialog
- GUI: NLS improvements
- BIOS: fixed floppy boot menu
- BIOS: expose the VM UUID in the DMI/SMBIOS area
- VGA: fixed CGA video modes
- VGA: fixed 8-bit DAC handling (Solaris setup)
- VMM: fixed issue with VT-x on Windows 64-bit hosts
- VMM: improved compatibility with Linux KVM
- VMM: fixed issues with Fedora 8 guests
- VMM: fixed fatal errors while installing Windows guests when using AMD-V
- VMM: fixed sporadic hangs when minimizing VM window and using VT-x/AMD-V

- VMM: fixed high load of ksoftirq on tickless Linux hosts
- VMM: fixed Windows 2000 guests hangs related to IRQ sharing
- VMM: fixed sporadic errors during openSUSE 10.3 installation
- VMM: fixed issue with Linux 2.6.23 guests
- VMM: fixed issues with Solaris guests
- VMM: fixed stability issue related to incorrect relocations
- Serial: significantly reduced CPU utilization
- Network: fixed issues with FreeBSD guests
- Network: added MII support (100MBit detection fix)
- Network: improved MAC address handling
- Network: added PXE release logging
- IDE: large reads from CD could exceed the I/O buffer size
- Audio: load ALSA dynamically on Linux (i.e. do not fail when ALSA is not present)
- VRDP: support additional RDP clients (SunRay, WinConnect, Mac OS X)
- VRDP: fixed issues when client color depth is higher than server color depth
- VRDP: make PAM authentication service name configurable
- VRDP: increased stack size to deal with stack consuming PAM library calls
- Additions: various fixes and enhancements to clipboard handling
- Windows Additions: fixed issues with Additions on NT 4 guests
- Windows Additions: added support for 8-bit video modes
- Windows Additions: allow specifying custom resolutions for secondary screens
- Windows Additions: several fixes and improvements for DirectDraw
- Windows Additions: improved the mouse filter driver compatibility with other mouse drivers
- Linux Additions: several fixes and enhancements to Shared Folders
- Linux Additions: added support for X.org Server 1.4
- Shared folders: fixed MS Powerpoint access issues (Linux host)

- API: fixed RPC_E_CHANGED_MODE startup error on Windows hosts
- API: fixed SMP race condition on Linux hosts
- API: fixed stability issues on Windows hosts in low memory conditions

12.6 Version 1.5.0 (2007-08-31)

As major new features, Version 1.5 adds:

- Seamless windows (see chapter 4.6, *Seamless windows*, page 60)
- Virtual serial ports (see chapter 3.7.9, *Serial ports*, page 50)
- Support for 64-bit Windows hosts (see chapter 1.3.1, *Supported host operating systems*, page 12)
- Intel PXE 2.1 network boot
- Guest Additions for IBM OS/2 Warp

In addition, the following items were fixed and/or added:

- GUI: sometimes two mouse cursors were visible when Windows guest additions became active
- GUI: added VT-x/AMD-V settings
- GUI: disable 'Show log...' menu entry to prevent crash if VM list is empty
- GUI: the log window grabbed the keyboard
- GUI: fixed error handling if Linux host clipboard initialization fails
- GUI: pass the Pause key and the PrtScrn key to the guest (Linux hosts)
- GUI: increased maximum guest RAM to 2 GB (Windows host)
- GUI: improved rendering performance (Windows host)
- GUI: status lights for USB and shared folders
- GUI: properly respect the DISPLAY environment variable
- GUI: download Guest Additions from virtualbox.org in case they are not present locally
- VRDP: support for multimonitor configurations in Windows guests
- VRDP: support for MS RDP6 and MS RDP Mac clients

- VRDP: added support for WinConnect RDP client
- VRDP: performance improvements
- VRDP: fixed sporadic client disconnects
- VBoxManage: never delete existing target during clonevdi
- VBoxManage: properly print the size of currently used hard disks
- VMM: fixed Xandros Desktop 4.1 hang
- VMM: fixed VT-x/AMD-V hang with newer versions of gcc (Linux hosts)
- VMM: improved stability of VT-x
- VMM: check for disabled AMD-V when detecting support
- VMM: fixed AMD-V issue when running OS/2 guests
- VMM: fixed application startup regressions (e.g. VideoReDo)
- VMM: fixed regression that broke disk access in OS/2 and OpenBSD guests (possibly much more)
- VMM: fixed crashes if memory allocation failed (Linux)
- VMM: fixed enabling of Local APIC on AMD hosts (fixed Ubuntu Feisty installation kernel hang during boot)
- VMM: fixed XFree86 4.3 (Debian/Sarge) segfaults when switching to text mode
- VMM: refuse to start when KVM is active (Linux Host)
- VMM: fixed bootup hangs with ReactOS
- VMM: fixed out-of-memory errors under certain environments with enough appropriate memory available
- API: fixed occassional crashes of the VBoxSVC server during VM shutdown (Linux host)
- API: some components were not notified when mounting a CD/DVD
- VMDK: improve geometry compatibility with existing VDMK images
- IDE/Floppy: optionally make non-available host device non-fatal
- IDE: improve emulation accuracy of the IRQ line between master and slave drive
- IDE: guest could freeze when unmounting the CD/DVD drive
- VGA: several text mode fixes in particular with Windows DOS boxes

- USB: fixed some issues with Windows hosts
- USB: fixed race condition between udev and USB filters (Linux host)
- Shared Folders: reversed network provider order to increase mapping performance (Windows guest)
- Shared Folders: browsable from Windows Explorer (Windows guests)
- Shared Folders: stability fixes (Windows guest)
- Shared Folders: case sensitivity fixes (Windows guest and Linux host)
- Audio: fall back to the NULL audio driver if no voice could be opened
- NAT: fixed crash
- Guest Additions: reworked the shared clipboard for Linux hosts and guests based on user feedback about problems with individual applications
- Guest Additions: don't allow to disable mouse pointer integration for Linux guests as an Xorg hardware mouse cursor cannot be turned into a software mouse cursor
- Guest Additions: Linux guests shipping Xorg 1.3 (e.g. Fedora 7, Ubuntu Gutsy) are now supported
- Guest Additions: added DirectDraw support to the Windows display driver

12.7 Version 1.4.0 (2007-06-06)

- General: added support for OS X hosts
- General: added support for AMD64 hosts
- General: signed all executables and device drivers on Windows
- GUI: added user interface for Shared Folders
- GUI: added context menu for network adapters
- GUI: added VM description field for taking notes
- GUI: always restore guest mouse pointer when entering VM window (Windows host)
- GUI: added configuration options for clipboard synchronization
- GUI: improved keyboard handling on Linux hosts
- GUI: added first run wizard

- GUI: improved boot device order dialog
- GUI: auto-resize did not work after save/restore
- GUI: restore original window size when returning from fullscreen mode
- GUI: fixed screen update when switching to fullscreen mode
- GUI: the size of the VM window was sometimes resetted to 640x480
- GUI: added localizations
- GUI: fixed size report of ISO images greater than 4GB
- GUI: various minor improvements
- VBoxManage: added convertdd command
- API: automatically start and terminate VBoxSVC on Linux and OS X hosts
- VMM: increased startup performance due to lazy memory allocation
- VMM: significantly increased maximum guest memory size
- VMM: fixed issues with V86 mode
- VMM: support V86 extensions (VME)
- VMM: support guests with a full GDT
- VMM: fixed boot hangs for some Linux kernels
- VMM: improved FreeBSD and OpenBSD support
- VMM: improved performance of guests that aggressively patch kernel code (very recent Linux 2.6 kernels)
- VMM: added workaround for a design flaw in AMD AM2 CPUs where the timestamp counter shows large differences among CPU cores
- VMM: fixed Linux guests with grsecurity
- VMM: fixed issue on 2G/2G Linux kernels (even 1G/3G kernels should work)
- VMM: fixed Linux detection of Local APIC on non-Intel and non-AMD CPUs
- VMM: timing improvements with high host system loads (VM starvation)
- VMM: experimental AMD SVM hardware virtualization support now also handles real and protected mode without paging
- VMM: added system time offset parameter to allow for VMs to run in the past or future

- VMM: provide an MPS 1.4 table if the IOAPIC is enabled
- VRDP: allow binding the VRDP server to a specific interface
- VRDP: added support for clipboard synchronization
- VRDP: fixed problems with OS X RDP client
- VRDP: added support for multiple simultaneous connections to one VM
- VRDP: added support for MS RDP6 clients (Vista)
- Storage: experimental support for VMDK images (writethrough mode only, no snapshots yet)
- Storage: raw host disk support, including individual partitions
- IDE: improve CHS geometry detection
- IDE: fixed problem that only one VM could open an immutable image
- NAT: allow more than one card configured for NAT networking
- NAT: pass first entry in DNS search list (Linux host) or primary DNS suffix (Windows host) as domain name in DHCP
- NAT: support UDP broadcasts, which enables using Windows shares
- NAT: only warn if the name server could not be determined, no fatal error anymore
- NAT: fix a potential problem with incorrect memory allocation
- Internal Networking: fixed issue on Windows hosts
- Host Interface Networking: fixed sporadic crashes on interface creation/destruction (Windows host)
- Host Interface Networking: reworked TAP handling for Linux 2.6.18+ compatibility
- PXE: show error for unsupported V86 case
- PXE: small fix for parsing PXE menu entry without boot server IP
- Network: fixed network card hang after save/restore
- USB: rewrote Windows USB handling without the need for a filter driver
- USB: possible to steal arbitrary devices in Windows
- Serial: added serial ports with support for named pipes (local domain sockets) on the host

- Audio: fixed problem with ALSA on Linux before 2.6.18 blocking other ALSA clients on the system
- Audio: fixed problem with ALSA on AMD64 hosts
- Input: fixed PS/2 mouse detection in Win 3.x guests
- Shared Folders: fixed VM save/restore behaviour
- Shared Folders: functionality and stability fixes
- Shared Folders: allow non admin users to map folders
- Additions: added clipboard synchronization
- Windows Additions: fixed dynamic resolution changes after save/restore
- Windows Additions: added AMD PCNet driver for Windows Vista guests (with kind permission from AMD)
- Linux Additions: fixed a dependency problem which caused the vboxadd kernel module sometimes start after the X server
- Linux Additions: make VBox version visible in Linux modules with modinfo
- Linux Additions: make X11 guest video driver accept arbitrary X resolutions
- Linux Additions: make X11 setup work if /tmp uses a separate file system
- Linux Additions: better support unknown distributions
- Linux Installer: force a non-executable stack for all binaries and shared libraries
- Linux Installer: make it work on SELinux-enabled systems
- Linux Installer: ship VBoxTunctl

12.8 Version 1.3.8 (2007-03-14)

- Windows installer: fixed installation problem if UAC is active
- Linux installer: added RPM for rhel4 and Mandriva 2007.1
- Linux installer: remove any old vboxdrv modules in /lib/modules/*/misc
- Linux installer: many small improvements for .deb and .rpm packages
- Linux installer: improved setup of kernel module
- GUI: Host-Fn sends Ctrl-Alt-Fn to the guest (Linux guest VT switch)
- GUI: fixed setting for Internal Networking

- GUI: show correct audio backend on Windows (dsound)
- GUI: improved error messages if the kernel module is not accessible
- GUI: never fail to start the GUI if the kernel module is not accessible
- VMM: fixed occasional crashes when shutting down Windows TAP device
- VMM: fixed issues with IBM's 1.4.2 JVM in Linux guests
- VRDP: fixed color encoding with 24bpp
- BIOS: zero main memory on reboot
- BIOS: added release logging
- USB: fixed parsing of certain devices to prevent VBoxSVC crashes
- USB: properly wakeup suspended ports
- USB: fixed a problem with unplugged USB devices during suspend
- Audio: fixed crashes on Vista hosts
- NAT: allow configuration of incoming connections (aka port mapping)
- Network: hard reset network device on reboot
- iSCSI: fixed a hang of unpaused VMs accessing unresponsive iSCSI disks
- Linux Additions: support Xorg 7.2.x
- Linux Additions: fixed default video mode if all other modes are invalid
- Linux Additions: set default DPI to 100,100
- Linux Additions: fixed initialization of video driver on X server reset

12.9 Version 1.3.6 (2007-02-20)

- Windows installer: perform installation for all users instead of just the current user (old behavior still available)
- Linux installer: fixed license display to not block installation
- Linux installer: added RPM for openSUSE 10.2
- GUI: fixed problems with several keyboard layouts on Linux hosts
- GUI: added online help on Linux hosts (using kchmviewer)
- GUI: fixed handle leak on Windows hosts

- Graphics: increased VRAM limit to 128MB
- BIOS: fixed CD/DVD-ROM detection in Windows Vista guests
- VMM: fixed incompatibility with OpenBSD 4.0
- VDI: fixed issues with snapshot merging
- Network: fixed incompatibility between Vista UAC and Host Interface Networking
- Network: fixed issues with Windows NT 4.0 guests
- Audio: fixed problem with ALSA on Linux before 2.6.18 causing system reboots
- VRDP: added support for MS RDP 6.0 clients
- VRDP: fixed issue with PAM authentication on certain distributions
- VRDP: fixed sporadic disconnects with MS RDP clients
- iSCSI: improved behavior when pausing a VM with iSCSI connections
- iSCSI: improved read timeout handling

12.10 Version 1.3.4 (2007-02-12)

- General: fixed unresolved symbol issue on Windows 2000 hosts
- General: added warnings at VirtualBox startup when there is no valid Linux kernel module
- General: fixed problem with unrecognized host CDROM/DVD drives on Linux
- General: fixed compatibility issue with SELinux
- GUI: improved USB user interface, easier filter definitions, menu to directly attach specific devices
- GUI: added VM settings options for VRDP
- GUI: fixed GDI handle leak on Windows hosts
- GUI: worked around issue in the Metacity window manager (GNOME) leading to unmovable VM windows
- GUI: show an information dialog before entering fullscreen mode about how to get back
- GUI: several fixes and improvements

- VMM: fixed occasional crashes when shutting down a Windows guest
- VMM: fixed crash while loading Xorg on openSUSE 10.2
- VMM: fixed problems with OpenBSD 3.9 and 4.0
- VMM: fixed crash while loading XFree86 in SUSE 9.1
- VMM: fixed Debian 3.1 (Sarge) installation problem (network failure)
- VMM: fixed crash during SUSE 10.2 installation
- VMM: fixed crash during Ubuntu 7.04 RC boot
- VMM: fixed crash during ThinClientOS (Linux 2.4.33) bootup
- ATA/IDE: pause VM when host disk is full and display message
- ATA/IDE: fixed incompatibility with OpenSolaris 10
- VDI containers: do not allocate blocks when guest only writes zeros to it (size optimization when zeroing freespace prior to compacting)
- CDROM/DVD: fixed media recognition by Linux guests
- Network: corrected reporting of physical interfaces (fixes Linux guest warnings)
- Network: fixed IRQ conflict causing occassional major slowdowns with XP guests
- Network: significantly improved send performance
- Audio: added mixer support to the AC'97 codec (master volume only)
- Audio: added support for ALSA on Linux (native, no OSS emulation)
- iSCSI: improved LUN handling
- iSCSI: fixed hang due to packet overflow
- iSCSI: pause VM on iSCSI connection loss
- Linux module: never fail unloading the module (blocks Ubuntu/Debian uninstall)
- Linux module: improved compatibility with NMI watchdog enabled
- Windows Additions: fixed hardware mouse pointer with Windows 2003 Server guests
- Linux Additions: compile everything from sources instead of using precompiled objects
- Linux Additions: better compatibility with older glibc versions

- Linux Additions: when uninstalling, only delete the files we put there during installation, don't remove the directory recursively to prevent unwanted data loss
- Linux Installer: added support for Slackware
- Linux Additions: added support for Linux 2.4.28 to 2.4.34
- VRDP: fixed sporadic disconnects with MS RDP clients
- VRDP: fixed race condition during resolution resize leading to rare crashes

12.11 Version 1.3.2 (2007-01-15)

- General: added experimental support for Windows Vista as a host
- General: added support for Windows Vista as a guest
- GUI: numerous improvements including a redesigned media manager
- BIOS: added DMI information for recent Linux kernels
- VMM: experimental support for AMD SVM hardware virtualization extensions
- VMM: significant performance improvements for Linux 2.6 guests
- VMM: performance improvements for Windows guests
- Network: fixed issues with DOS guests
- Network: fixed creation of more than one host interface during process lifetime on Windows
- VBoxManage: added support for compacting VDI files (requires zeroing freespace in the guest)
- API: startup even when a VM configuration file is inaccessible or corrupted
- API: faster startup using lazy media access checking
- Linux Additions: fixed several installation issues and added better error checks
- Linux Additions: added support for X.org 7.1
- Installer: added packages for Ubuntu 6.10 (Edgy Eft), Ubuntu 6.06 LTS (Dapper Drake) and Debian 4.0 (Etch)

12.12 Version 1.2.4 (2006-11-16)

Several bug fixes that accidentially didn't make it into 1.2.2

12.13 Version 1.2.2 (2006-11-14)

Note: Guest Additions have to be updated for the enhanced VRDP features to work.

- Linux Additions: improved compatibility with Red Hat distributions
- Linux Additions: enhanced display performance, solved several issues
- Linux Additions: added color pointer support
- Linux Additions: added support for X.org 7.x
- VMM: fixed sporadic mouse reset problem
- VMM: fixed several issues with Linux guests
- VMM: significant performance improvements for Linux 2.6 guests
- VMM: significant general performance improvements
- VMM: fixed sporadic reboot problems (logo hang)
- VMM: added support for Intel VT-x (aka Vanderpool)
- VMM: experimental support for IBM OS/2 Warp (requires VT-x to be enabled)
- USB: added support for isochronous transfers (webcams, audio, etc.)
- USB: fixed problem with devices not showing up after a guest reboot
- USB: fixed several issues
- BIOS: fixed use of fourth boot device
- BIOS: added boot menu support
- BIOS: added support for disks up to 2 Terabytes
- VRDP: significantly enhanced performance and reduced bandwidth usage through new acceleration architecture
- VBoxManage: added support for capturing network traffic
- GUI: added fullscreen mode
- GUI: fixed several problems

12.14 Version 1.1.12 (2006-11-14)

- Additions: enabled more display modes for X.org 7.x
- VMM: stability improvements
- VMM: resolved excessive performance degradation caused by Symantec Antivirus
- iSCSI: fixed memory corruption issue
- VBoxSDL: made hostkey configurable
- VRDP: report error in case binding to the port fails
- VRDP: added mouse wheel support
- NAT: significant performance improvements
- Network: stability fixes
- Network: significant performance improvements
- ACPI: improved host power status reporting
- PXE: added support for Microsoft RIS / ProxyDHCP
- PXE: fixed several issues, added diagnostic messages

12.15 Version 1.1.10 (2006-07-28)

- IDE: added workaround for Acronis TrueImage (violates IDE specification)
- IDE: resolved issues with certain Linux guests
- ACPI: further improved host power status reporting
- API: fixed several race conditions and improved reliability
- API: increased maximum guest RAM size to 2GB (Linux host) and 1.2GB (Windows host)
- USB: added option to set the OHCI timer rate
- VMM: fixed several issues
- VRDP: fixed infinite resize loop
- GUI: changed the default host key to Right Control

12.16 Version 1.1.8 (2006-07-17)

- IDE: new ATA implementation with improved performance, reliability and better standards compliance
- IDE: added experimental support for ATAPI passthrough (to use CD/DVD burners inside VMs)
- VMM: fixed user mode IOPL handling (hwclock failure)
- VMM: fixed crashes upon termination in Linux X servers
- VMM: fixed problems with Knoppix 5.0 (and other Linux kernels 2.6.15+)
- VMM: improved handling of self modifying code (aka Linux 2.6.15+ errors)
- VMM: introduce release logging for better servicability
- VMM: significant performance improvements, especially for Linux 2.6 guests
- VRDP: several issues have been fixed
- VRDP: fixed enhanced rdesktop to build correctly under Linux 2.6.15+
- Additions: added support for SUSE 10.1 and Fedora Core 5
- NAT: improved performance and stability
- NAT: handle host IP configuration changes at runtime
- VBoxManage: made VRDP authentication configurable
- VDI: added workaround against possible Windows host deadlocks caused by a sychronisation flaw in Windows
- ACPI: improved host power status reporting

12.17 Version 1.1.6 (2006-04-18)

- ACPI: added workaround for XP SP2 crash in intelppm.sys (the real problem is a bug in this driver)
- IDE: added support for image files of up to 8 terabytes
- API: fixed several race conditions on SMP systems
- Network: significant performance improvements
- VRDP: fixed several issues with USB redirection

- IDE: added workaround for Windows 2000 installation problems due to a bug in the Windows disk driver (see troubleshooting section)
- VRDP: provide extensive connection information (also exposed through VBoxManage)
- Linux module: added support for Linux 2.6.16
- VBoxManage: improved support for immutable disk images
- iSCSI: several fixes
- USB: several fixes
- VBoxSDL: added switch for fixed video mode and guest image centering
- VMM: improved performance of Linux 2.6.x guests

12.18 Version 1.1.4 (2006-03-09)

Note: The configuration file format has been changed. After applying this update, execute "VBoxManage updatesettings" to convert your configuration to the new format.
Note: Guest Additions have to be updated.

- General: added support for multi-generation snapshots
- VMM: fixed Linux guest reboot regression
- VRDP: added client authentication through external authentication libraries (WinLogon and PAM interfaces are provided as sample code)
- VRDP: close TCP connection immediately when receiving bad data from the remote side
- VRDP: improved Microsoft RDP client support
- XPCOM: fixed race condition on SMP systems that could lead to hung client processes (Linux host)
- API: fixed race condition on SMP systems
- Network: added AMD PC-Net II 100MBit network card (Am79C973)
- Network: added PXE boot ROM for network boot
- Audio: fixed regression with Windows 2000 guests
- Audio: pause playback when VM is paused
- iSCSI: added standards compliant iSCSI initiator for transparent access of iSCSI targets

- VBoxSDL: ship on Windows as well
- VBoxManage: added command to clone a VDI file to another one having a different UUID
- Additions: added Linux additions (timesync, mouse pointer integration and graphics driver)
- Additions: added Shared Folders for Windows guests (except NT)
- Linux module: fixed compilation problem on SUSE 10 system
- Linux installer: added custom shell script installer

12.19 Version 1.1.2 (2006-02-03)

Note: Guest Additions have to be updated. The installation method has changed.

- BIOS: fixed CMOS checksum calculation (to avoid guest warnings)
- BIOS: improved APM support (to avoid guest warnings)
- IDE: Linux 2.6.14+ and OpenBSD now operate the controller in UDMA mode by default
- VMM: fixed hang when rebooting Windows 2000 guests with enabled audio adapter
- VMM: fixed random user mode crashes with OpenBSD guests
- VMM: increased timing accuracy (PIT, RTC), reduced PIT query overhead
- VMM: tamed execution thread to make GUI more responsive (esp. when executing real mode guest code such as bootloaders)
- VMM: significant performance enhancements for OpenBSD guests
- VMM: several performance enhancements
- VMM: improved memory layout on Windows hosts to allow for large amounts of guest RAM
- VMM: significantly improved VM execution state saving and restoring (at the expense of state file sizes)
- ACPI: fixed Windows bluescreen when assigning more than 512MB RAM to a guest
- ACPI: correctly report battery state when multiple batteries are present on the host (Linux hosts)

- ACPI: enabled by default for newly created VMs
- APIC: added optional I/O APIC
- Graphics: fixed distortion when changing guest color depth without changing the resolution
- VRDP: added support for remote USB (requires special rdesktop client)
- VRDP: added support for the Microsoft RDP client
- VRDP: improved audio support
- Floppy: controller can be disabled
- Floppy: fixed “no disk in drive” reporting
- Floppy: fixed writing to floppy images
- VBoxManage: restructured USB device filter syntax to make it more intuitive
- VBoxManage: added command for setting guest logon credentials
- Additions: added installer for Windows 2000/XP/2003 guests
- Additions: added custom GINA module which hooks MSGINA and can perform automatic logons using credentials retrieved from the VMM
- Documentation: added draft of VirtualBox user manual

12.20 Version 1.0.50 (2005-12-16)

Note: Guest Additions have to be updated

- VMM: added support for OpenBSD guests
- VMM: fixed a memory leak
- Network: added Internal Networking (to directly wire VMs without using host interfaces and making the traffic visible on the host)
- Network: fixed crash/hang at exit with TAP on Linux
- Graphics: added support for additional custom VESA modes
- Graphics: added support for VESA modes with y offset
- VRDP: added support for remote audio (PCM encoding)
- USB: fixed several potential crashes
- USB: fixed revision filter matching
- USB: fixed support for devices with integrated USB hubs

12.21 Version 1.0.48 (2005-11-23)

Note: The configuration has to be deleted as the format has changed. On Linux, issue rm -rf ~/.VirtualBox. On Windows, remove the directory C:\Documents and Settings\<username>\.VirtualBox. If you fail to do so, VirtualBox will not startup. Note: Guest Additions have to be updated

- VMM: fixed a Linux 2.6 guest panic on certain P4 CPUs
- VMM: performance improvements
- Graphics: fixed y offset handling in dynamic resolution mode (secure labeling support)
- VDI: added support for immutable independent images (part of the upcoming snapshot feature)
- Additions: added VBoxControl command line utility to get/set the guest video acceleration status
- Additions: video acceleration is turned off by default, use VBoxControl to enable it. It usually helps for VRDP performance.
- GUI: DirectDraw support for faster display handling on Win32.
- GUI: allow creation and assignment of disk images in the New VM wizard.
- USB: fixed high CPU load on certain Linux distributions
- VBoxSDL: fixed several secure labeling issues (crash at exit, protection against guest video modes greater than what SDL provides on the host)
- VBoxManage: convert command line parameters from the current codepage to Unicode

12.22 Version 1.0.46 (2005-11-04)

Note: Guest Additions have to be updated

- Linux: VirtualBox binaries can now be started from directories other than the installation directory
- VMM: added support for PAE guest mode
- VMM: added support for hosts running in NX (No Execute) / DEP (Data Execution Prevention) mode
- Graphics: fixes for dynamic resolution handling

- Linux module: yet another kernel panic fix due to weird patches in RedHat Enterprise Linux 4 Update 2
- VBoxSVC: if VBOX_USER_HOME is set, look for configuration in this directory (default: $HOME/.VirtualBox)

12.23 Version 1.0.44 (2005-10-25)

Note: Guest Additions have to be updated.

- Installer: greatly improved Windows installer, fixed uninstall and perform driver and COM registration through MSI
- VBoxManage: added commands to create and delete Win32 Host Interface Networking adapters
- VDI: updated virtual disk image format (for newly created images; old images continue to work) with enhanced write performance and support for the upcoming snapshot feature
- Network: performance improvements
- Graphics: added hardware acceleration to virtual graphics adapter and corresponding Guest Additions driver
- Graphics/Additions/GUI: added dynamic resizing support
- Graphics: added workaround for buggy VESA support in Windows Vista/Longhorn
- VRDP: performance and stability improvements; added support for graphics acceleration architecture
- USB: restructured USB subsystem; added support for filters to autocapture devices that meet defined criteria
- GUI: added mouse wheel support
- VMM: added support for PAE host mode

12.24 Version 1.0.42 (2005-08-30)

Note: The configuration has to be deleted as the format has changed. On Linux, issue rm -rf ~/.VirtualBox. On Windows, remove the directory C:\Documents and Settings\<username>\.VirtualBox. If you fail to do so, VirtualBox will not startup. Note: Guest Additions have to be updated.

- USB: added USB support for Windows hosts

- Network: renamed TUN to “Host Interface Networking” and TAP on Linux
- Network: added support for Host Interface Networking on Windows hosts
- Network: added “cable connected” property to the virtual network cards
- Floppy: added a virtual floppy drive to the VM and support for attaching floppy images and capturing host floppy drives
- DVD/CD: added host CD/DVD drive support
- BIOS: added boot order support
- Saved states: made location configurable (default, global setting, machine specific setting, including VBoxManage command support)
- VMM: added support for host CPUs without FXSR (e.g. Via Centaur)
- VMM: increased performance of Linux 2.6 guests
- VMM: improved timing
- VMM: fixed traps in XP guests with ACPI enabled
- VBoxManage: added remote session start function (tstHeadless has been removed from the distribution)
- VBoxManage: restructured commands, added numerous improvements
- GUI: propagate hostkey change to all running instances
- GUI: perform image access tests asynchronously
- GUI: added boot order support
- GUI: user interface redesign

12.25 Version 1.0.40 (2005-06-17)

Note: The configuration has to be deleted as the format has changed. On Linux, issue rm -rf ~/.VirtualBox. On Windows, remove the directory C:\Documents and Settings\<username>\.VirtualBox. If you fail to do so, VirtualBox will not startup. Note: Guest Additions have to be updated.

- SDK: ship VirtualBox development tools and sample program
- BIOS: made startup logo animation configurable for OEM customers
- BIOS: fixed network card detection under DOS
- Graphics: fixed VESA modes in XP and XFree86/X.org

- Network: fixed Linux guest issues
- Network: fixed NAT DHCP server to work with MS-DOS TCP/IP
- Network: fixed performance issue under heavy guest CPU load
- Network: fixed errors with more than one network card
- USB: added experimental USB support for Linux hosts
- VMM: fixed DOS A20 gate handling in real mode
- VMM: fixed TSS IO bitmap handling (crash in Debian/Knoppix hardware detection routine)
- VMM: fixed IO issue which broke VESA in X11
- VMM: performance improvements for Linux guests
- VMM: added local APIC support
- VBoxSDL: added pointer shape support and use host pointer in fullscreen mode if available
- GUI: determine system parameters (e.g. maximum VDI size) using the API
- GUI: added detailed error information dialogs
- GUI: special handling of inaccessible media
- API: better error message handling, provide system parameters, handle inaccessible media
- Guest Additions: implemented full pointer shape support for all pointer color depths including alpha channel
- VBoxManage: several command extensions

12.26 Version 1.0.39 (2005-05-05)

Note: Guest Additions have to be updated.

- Linux: converted XPCOM runtime to a single shared object
- Linux: fixed SIGALRM process crash on certain distributions
- VMM: fixed Linux guests with grsecurity (address space scrambling)
- ACPI: added experimental ACPI support
- VRDP: added shadow buffer for reduced bandwidth usage

- VRDP: added support for pointer shapes and remote pointer cache
- GUI: added support for pointer shapes
- Windows Additions: added support for high resolution video modes, including multi screen modes (2, 3 and 4 screens)
- VBoxManage: added new command line tool to automate simple administration tasks without having to write application code

12.27 Version 1.0.38 (2005-04-27)

- GUI: fixed creation of disk images larger than 4GB
- GUI: added network and audio configuration panels
- GUI: several keyboard issues fixed
- VBoxSDL: fixed -tunfd handling and added -tundev (Linux host)
- IDE: significant performance improvements in DMA modes
- Video: VRAM size is now configurable (1MB - 128MB; default 4MB)
- VMM: fixed several crashes and hangs while installing certain builds of Windows 2000 and XP
- VMM: allow guests to have more than 512MB of RAM
- VMM: resolved compatibility issues with SMP systems (Windows Host)
- VRDP: process cleanup on Linux fixed
- Linux module: fixed build error on Red Hat 2.4.21-15-EL
- NT Additions: fixed installation and a trap
- Win2k/XP Additions: fixed installation

12.28 Version 1.0.37 (2005-04-12)

Initial build with change log.

13 Known issues

The following section describes some issues that are known not to work in VirtualBox 1.6.2. Some of these issues will be fixed in later releases.

- **NAT mode.** A virtual network adapter configured for NAT is slower than an adapter configured for host interface networking.
- **Vista 64 bits host** :
  - Known stability issues when using USB
- **Mac OS X host.** The following restrictions apply (all of which will be resolved in future versions):
  - No support for Host Interface Networking
  - No support for Internal Networking
  - No support for audio input
  - No support for VT-x/AMD-V (rarely required)
  - No support for raw disk access
  - The numlock emulation isn't implemented yet
  - No ACPI information (battery status, power source) is reported to the guest
  - The VirtualBox kernel extension is currently accessible from all user accounts
- **Solaris hosts.** The following restrictions apply (all of which will be resolved in future versions):
  - There are some limitations on host interface networking (see chapter 6.8, *Host Interface Networking and bridging on OpenSolaris hosts*, page 83)
  - Audio playback is experimental, and audio input is not yet supported
  - No support for using USB devices on the host
  - No ACPI information (battery status, power source) is reported to the guest
- **Guest Additions for Solaris.** Shared folders are not yet supported with Solaris guests.

- **Guest Additions for OS/2.** Shared folders are not yet supported with OS/2 guests yet. In addition, seamless windows and automatic guest resizing will probably never be implemented due to inherent limitations of the OS/2 graphics system.
- **Linux guests and AMD Barcelona CPUs.** Most Linux based guests will fail with AMD Phenoms or Barcelona-level Opterons due to a bug in the Linux kernel. Enable the IO-APIC to work around the problem.

14 Third-party licenses

VirtualBox incorporates materials from several Open Source software projects. Therefore the use of these materials by VirtualBox is governed by different Open Source licenses. This document reproduces these licenses and provides a list of the materials used and their respective licensing conditions. Section 1 contains a list of the materials used. Section 2 reproduces the applicable Open Source licenses. For each material, a reference to its license is provided.

14.1 Materials

- VirtualBox contains portions of QEMU which is governed by licenses chapter 14.2.1, *X Consortium License (X11)*, page 183 and chapter 14.2.2, *GNU Lesser General Public License (LGPL)*, page 184 and

  (C) 2003-2005 Fabrice Bellard Copyright (c) 2004-2005 Vassili Karpov (malc) Copyright (c) 2004 Antony T Curtis Copyright (c) 2003 Jocelyn Mayer

- VirtualBox contains code which is governed by license chapter 14.2.1, *X Consortium License (X11)*, page 183 and

  Copyright 2004 by the Massachusetts Institute of Technology.

- VirtualBox contains code of the BOCHS VGA BIOS which is governed by license chapter 14.2.2, *GNU Lesser General Public License (LGPL)*, page 184 and

  Copyright (C) 2001, 2002 the LGPL VGABios developers Team.

- VirtualBox contains code of the BOCHS ROM BIOS which is governed by license chapter 14.2.2, *GNU Lesser General Public License (LGPL)*, page 184 and

  Copyright (C) 2002 MandrakeSoft S.A. Copyright (C) 2004 Fabrice Bellard Copyright (C) 2005 Struan Bartlett.

- VirtualBox contains the zlib library which is governed by license chapter 14.2.3, *zlib license*, page 192 and

  Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler.

- VirtualBox may contain OpenSSL which is governed by license chapter 14.2.4, *OpenSSL license*, page 192 and

  Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com).

- VirtualBox may contain NSPR and XPCOM which is governed by license chapter 14.2.5, *Mozilla Public License (MPL)*, page 193 and

  Copyright (C) The Authors.

- VirtualBox contains Slirp which is governed by license chapter 14.2.6, *Slirp license*, page 202 and was written by Danny Gasparovski.

  Copyright (c), 1995,1996 All Rights Reserved.

- VirtualBox contains liblzf which is governed by license chapter 14.2.7, *liblzf license*, page 202 and

  Copyright (c) 2000-2005 Marc Alexander Lehmann <schmorp@schmorp.de>

- VirtualBox may ship with a modified copy of rdesktop which is governed by license chapter 14.2.8, *GNU General Public License (GPL)*, page 203 and

  Copyright (C) Matthew Chapman and others.

- VirtualBox may ship with a copy of kchmviewer which is governed by license chapter 14.2.8, *GNU General Public License (GPL)*, page 203 and

  Copyright (C) George Yunaev and others.

- VirtualBox contains Etherboot which is governed by license chapter 14.2.8, *GNU General Public License (GPL)*, page 203 with the exception that aggregating Etherboot with another work does not require the other work to be released under the same license (see `http://etherboot.sourceforge.net/clinks.html`). Etherboot is

  Copyright (c) Etherboot team.

- VirtualBox may contain code from Wine which is governed by license chapter 14.2.2, *GNU Lesser General Public License (LGPL)*, page 184 and

  Copyright 1993 Bob Amstadt, Copyright 1996 Albrecht Kleine, Copyright 1997 David Faure, Copyright 1998 Morten Welinder, Copyright 1998 Ulrich Weigand, Copyright 1999 Ove Koven

- VirtualBox contains code from lwIP which is governed by license chapter 14.2.9, *lwIP license*, page 208 and

  Copyright (c) 2001, 2002 Swedish Institute of Computer Science.

14.2 Licenses

14.2.1 X Consortium License (X11)

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

```
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

14.2.2 GNU Lesser General Public License (LGPL)

```
  GNU LESSER GENERAL PUBLIC LICENSE
      Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]

    Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
not price.  Our General Public Licenses are designed to make sure that
you have the freedom to distribute copies of free software (and charge
for this service if you wish); that you receive source code or can get
it if you want it; that you can change the software and use pieces of
it in new free programs; and that you are informed that you can do
these things.

  To protect your rights, we need to make restrictions that forbid
distributors to deny you these rights or to ask you to surrender these
rights.  These restrictions translate to certain responsibilities for
you if you distribute copies of the library or if you modify it.
```

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that

you conspicuously and appropriately publish on each copy an
appropriate copyright notice and disclaimer of warranty; keep intact
all the notices that refer to this License and to the absence of any
warranty; and distribute a copy of this License along with the
Library.

  You may charge a fee for the physical act of transferring a copy,
and you may at your option offer warranty protection in exchange for a
fee.

  2. You may modify your copy or copies of the Library or any portion
of it, thus forming a work based on the Library, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) The modified work must itself be a software library.

    b) You must cause the files modified to carry prominent notices
    stating that you changed the files and the date of any change.

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application. Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Library,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Library, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote
it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Library.

In addition, mere aggregation of another work not based on the Library
with the Library (or with a work based on the Library) on a volume of

a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the

Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for

reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues),

conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License. If you cannot
distribute so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you
may not distribute the Library at all. For example, if a patent
license would not permit royalty-free redistribution of the Library by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  12. If the distribution and/or use of the Library is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus
excluded. In such case, this License incorporates the limitation as if
written in the body of this License.

  13. The Free Software Foundation may publish revised and/or new
versions of the Lesser General Public License from time to time.
Such new versions will be similar in spirit to the present version,
but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library
specifies a version number of this License which applies to it and
"any later version", you have the option of following the terms and
conditions either of that version or of any later version published by
the Free Software Foundation. If the Library does not specify a
license version number, you may choose any version ever published by
the Free Software Foundation.

  14. If you wish to incorporate parts of the Library into other free
programs whose distribution conditions are incompatible with these,
write to the author to ask for permission. For software which is
copyrighted by the Free Software Foundation, write to the Free
Software Foundation; we sometimes make exceptions for this. Our
decision will be guided by the two goals of preserving the free status

of all derivatives of our free software and of promoting the sharing
and reuse of software generally.

    NO WARRANTY

  15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO
WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW.
EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR
OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY
KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE
LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME
THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN
WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY
AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU
FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR
CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE
LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING
RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A
FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF
SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH
DAMAGES.

     END OF TERMS AND CONDITIONS

14.2.3 zlib license

```
This software is provided 'as-is', without any express or implied
warranty.  In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu
```

14.2.4 OpenSSL license

```
 This package is an SSL implementation written
 by Eric Young (eay@cryptsoft.com).
```

```
The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as
the following conditions are aheared to.  The following conditions
apply to all code found in this distribution, be it the RC4, RSA,
lhash, DES, etc., code; not just the SSL code.  The SSL documentation
included with this distribution is covered by the same copyright terms
except that the holder is Tim Hudson (tjh@cryptsoft.com).

Copyright remains Eric Young's, and as such any Copyright notices in
the code are not to be removed.
If this package is used in a product, Eric Young should be given attribution
as the author of the parts of the library used.
This can be in the form of a textual message at program startup or
in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software
   must display the following acknowledgement:
   "This product includes cryptographic software written by
   Eric Young (eay@cryptsoft.com)"
   The word 'cryptographic' can be left out if the rouines from the library
   being used are not cryptographic related :-).
4. If you include any Windows specific code (or a derivative thereof) from
   the apps directory (application code) you must include an acknowledgement:
   "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

The licence and distribution terms for any publically available version or
derivative of this code cannot be changed.  i.e. this code cannot simply be
copied and put under another distribution licence
[including the GNU Public Licence.]
```

14.2.5 Mozilla Public License (MPL)

```
MOZILLA PUBLIC LICENSE
      Version 1.1
```

---

1. Definitions.

1.0.1. "Commercial Use" means distribution or otherwise making the Covered Code available to a third party.

1.1. "Contributor" means each entity that creates or contributes to the creation of Modifications.

1.2. "Contributor Version" means the combination of the Original Code, prior Modifications used by a Contributor, and the Modifications made by that particular Contributor.

1.3. "Covered Code" means the Original Code or Modifications or the combination of the Original Code and Modifications, in each case including portions thereof.

1.4. "Electronic Distribution Mechanism" means a mechanism generally accepted in the software development community for the electronic transfer of data.

1.5. "Executable" means Covered Code in any form other than Source Code.

1.6. "Initial Developer" means the individual or entity identified as the Initial Developer in the Source Code notice required by Exhibit A.

1.7. "Larger Work" means a work which combines Covered Code or portions thereof with code not governed by the terms of this License.

1.8. "License" means this document.

1.8.1. "Licensable" means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently acquired, any and all of the rights conveyed herein.

1.9. "Modifications" means any addition to or deletion from the substance or structure of either the Original Code or any previous Modifications. When Covered Code is released as a series of files, a Modification is:

A. Any addition to or deletion from the contents of a file containing Original Code or previous Modifications.

B. Any new file that contains any part of the Original Code or previous Modifications.

1.10. "Original Code" means Source Code of computer software code which is described in the Source Code notice required by Exhibit A as Original Code, and which, at the time of its release under this License is not already Covered Code governed by this License.

1.10.1. "Patent Claims" means any patent claim(s), now owned or hereafter acquired, including without limitation, method, process,

and apparatus claims, in any patent Licensable by grantor.

1.11. "Source Code" means the preferred form of the Covered Code for making modifications to it, including all modules it contains, plus any associated interface definition files, scripts used to control compilation and installation of an Executable, or source code differential comparisons against either the Original Code or another well known, available Covered Code of the Contributor's choice. The Source Code can be in a compressed or archival form, provided the appropriate decompression or de-archiving software is widely available for no charge.

1.12. "You" (or "Your") means an individual or a legal entity exercising rights under, and complying with all of the terms of, this License or a future version of this License issued under Section 6.1. For legal entities, "You" includes any entity which controls, is controlled by, or is under common control with You. For purposes of this definition, "control" means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. Source Code License.

2.1. The Initial Developer Grant.
The Initial Developer hereby grants You a world-wide, royalty-free, non-exclusive license, subject to third party intellectual property claims:

(a) under intellectual property rights (other than patent or trademark) Licensable by Initial Developer to use, reproduce, modify, display, perform, sublicense and distribute the Original Code (or portions thereof) with or without Modifications, and/or as part of a Larger Work; and

(b) under Patents Claims infringed by the making, using or selling of Original Code, to make, have made, use, practice, sell, and offer for sale, and/or otherwise dispose of the Original Code (or portions thereof).

(c) the licenses granted in this Section 2.1(a) and (b) are effective on the date Initial Developer first distributes Original Code under the terms of this License.

(d) Notwithstanding Section 2.1(b) above, no patent license is granted: 1) for code that You delete from the Original Code; 2) separate from the Original Code; or 3) for infringements caused by: i) the modification of the Original Code or ii) the combination of the Original Code with other software or devices.

2.2. Contributor Grant.
Subject to third party intellectual property claims, each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license

(a) under intellectual property rights (other than patent or trademark) Licensable by Contributor, to use, reproduce, modify,

display, perform, sublicense and distribute the Modifications created by such Contributor (or portions thereof) either on an unmodified basis, with other Modifications, as Covered Code and/or as part of a Larger Work; and

(b) under Patent Claims infringed by the making, using, or selling of Modifications made by that Contributor either alone and/or in combination with its Contributor Version (or portions of such combination), to make, use, sell, offer for sale, have made, and/or otherwise dispose of: 1) Modifications made by that Contributor (or portions thereof); and 2) the combination of Modifications made by that Contributor with its Contributor Version (or portions of such combination).

(c) the licenses granted in Sections 2.2(a) and 2.2(b) are effective on the date Contributor first makes Commercial Use of the Covered Code.

(d) Notwithstanding Section 2.2(b) above, no patent license is granted: 1) for any code that Contributor has deleted from the Contributor Version; 2) separate from the Contributor Version; 3) for infringements caused by: i) third party modifications of Contributor Version or ii) the combination of Modifications made by that Contributor with other software (except as part of the Contributor Version) or other devices; or 4) under Patent Claims infringed by Covered Code in the absence of Modifications made by that Contributor.

3. Distribution Obligations.

3.1. Application of License.
The Modifications which You create or to which You contribute are governed by the terms of this License, including without limitation Section 2.2. The Source Code version of Covered Code may be distributed only under the terms of this License or a future version of this License released under Section 6.1, and You must include a copy of this License with every copy of the Source Code You distribute. You may not offer or impose any terms on any Source Code version that alters or restricts the applicable version of this License or the recipients' rights hereunder. However, You may include an additional document offering the additional rights described in Section 3.5.

3.2. Availability of Source Code.
Any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License either on the same media as an Executable version or via an accepted Electronic Distribution Mechanism to anyone to whom you made an Executable version available; and if made available via Electronic Distribution Mechanism, must remain available for at least twelve (12) months after the date it initially became available, or at least six (6) months after a subsequent version of that particular Modification has been made available to such recipients. You are responsible for ensuring that the Source Code version remains available even if the Electronic Distribution Mechanism is maintained by a third party.

3.3. Description of Modifications.
You must cause all Covered Code to which You contribute to contain a file documenting the changes You made to create that Covered Code and the date of any change. You must include a prominent statement that the Modification is derived, directly or indirectly, from Original Code provided by the Initial Developer and including the name of the Initial Developer in (a) the Source Code, and (b) in any notice in an Executable version or related documentation in which You describe the origin or ownership of the Covered Code.

3.4. Intellectual Property Matters
(a) Third Party Claims.
If Contributor has knowledge that a license under a third party's intellectual property rights is required to exercise the rights granted by such Contributor under Sections 2.1 or 2.2, Contributor must include a text file with the Source Code distribution titled "LEGAL" which describes the claim and the party making the claim in sufficient detail that a recipient will know whom to contact. If Contributor obtains such knowledge after the Modification is made available as described in Section 3.2, Contributor shall promptly modify the LEGAL file in all copies Contributor makes available thereafter and shall take other steps (such as notifying appropriate mailing lists or newsgroups) reasonably calculated to inform those who received the Covered Code that new knowledge has been obtained.

(b) Contributor APIs.
If Contributor's Modifications include an application programming interface and Contributor has knowledge of patent licenses which are reasonably necessary to implement that API, Contributor must also include this information in the LEGAL file.

(c) Representations.
Contributor represents that, except as disclosed pursuant to Section 3.4(a) above, Contributor believes that Contributor's Modifications are Contributor's original creation(s) and/or Contributor has sufficient rights to grant the rights conveyed by this License.

3.5. Required Notices.
You must duplicate the notice in Exhibit A in each file of the Source Code. If it is not possible to put such notice in a particular Source Code file due to its structure, then You must include such notice in a location (such as a relevant directory) where a user would be likely to look for such a notice. If You created one or more Modification(s) You may add your name as a Contributor to the notice described in Exhibit A. You must also duplicate this License in any documentation for the Source Code where You describe recipients' rights or ownership rights relating to Covered Code. You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Code. However, You may do so only on Your own behalf, and not on behalf of the Initial Developer or any Contributor. You must make it absolutely clear than any such warranty, support, indemnity or liability obligation is offered by You alone, and You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the

Initial Developer or such Contributor as a result of warranty, support, indemnity or liability terms You offer.

3.6. Distribution of Executable Versions.
You may distribute Covered Code in Executable form only if the requirements of Section 3.1-3.5 have been met for that Covered Code, and if You include a notice stating that the Source Code version of the Covered Code is available under the terms of this License, including a description of how and where You have fulfilled the obligations of Section 3.2. The notice must be conspicuously included in any notice in an Executable version, related documentation or collateral in which You describe recipients' rights relating to the Covered Code. You may distribute the Executable version of Covered Code or ownership rights under a license of Your choice, which may contain terms different from this License, provided that You are in compliance with the terms of this License and that the license for the Executable version does not attempt to limit or alter the recipient's rights in the Source Code version from the rights set forth in this License. If You distribute the Executable version under a different license You must make it absolutely clear that any terms which differ from this License are offered by You alone, not by the Initial Developer or any Contributor. You hereby agree to indemnify the Initial Developer and every Contributor for any liability incurred by the Initial Developer or such Contributor as a result of any such terms You offer.

3.7. Larger Works.
You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License and distribute the Larger Work as a single product. In such a case, You must make sure the requirements of this License are fulfilled for the Covered Code.

4. Inability to Comply Due to Statute or Regulation.

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Code due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be included in the LEGAL file described in Section 3.4 and must be included with all distributions of the Source Code. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Application of this License.

This License applies to code to which the Initial Developer has attached the notice in Exhibit A and to related Covered Code.

6. Versions of the License.

6.1. New Versions.
Netscape Communications Corporation ("Netscape") may publish revised and/or new versions of the License from time to time. Each version will be given a distinguishing version number.

6.2. Effect of New Versions.
Once Covered Code has been published under a particular version of the License, You may always continue to use it under the terms of that version. You may also choose to use such Covered Code under the terms of any subsequent version of the License published by Netscape. No one other than Netscape has the right to modify the terms applicable to Covered Code created under this License.

6.3. Derivative Works.
If You create or use a modified version of this License (which you may only do in order to apply it to code which is not already Covered Code governed by this License), You must (a) rename Your license so that the phrases "Mozilla", "MOZILLAPL", "MOZPL", "Netscape", "MPL", "NPL" or any confusingly similar phrase do not appear in your license (except to note that your license differs from this License) and (b) otherwise make it clear that Your version of the license contains terms which differ from the Mozilla Public License and Netscape Public License. (Filling in the name of the Initial Developer, Original Code or Contributor in the notice described in Exhibit A shall not of themselves be deemed to be modifications of this License.)

7. DISCLAIMER OF WARRANTY.

COVERED CODE IS PROVIDED UNDER THIS LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, WARRANTIES THAT THE COVERED CODE IS FREE OF DEFECTS, MERCHANTABLE, FIT FOR A PARTICULAR PURPOSE OR NON-INFRINGING. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE COVERED CODE IS WITH YOU. SHOULD ANY COVERED CODE PROVE DEFECTIVE IN ANY RESPECT, YOU (NOT THE INITIAL DEVELOPER OR ANY OTHER CONTRIBUTOR) ASSUME THE COST OF ANY NECESSARY SERVICING, REPAIR OR CORRECTION. THIS DISCLAIMER OF WARRANTY CONSTITUTES AN ESSENTIAL PART OF THIS LICENSE. NO USE OF ANY COVERED CODE IS AUTHORIZED HEREUNDER EXCEPT UNDER THIS DISCLAIMER.

8. TERMINATION.

8.1. This License and the rights granted hereunder will terminate automatically if You fail to comply with terms herein and fail to cure such breach within 30 days of becoming aware of the breach. All sublicenses to the Covered Code which are properly granted shall survive any termination of this License. Provisions which, by their nature, must remain in effect beyond the termination of this License shall survive.

8.2. If You initiate litigation by asserting a patent infringement claim (excluding declatory judgment actions) against Initial Developer or a Contributor (the Initial Developer or Contributor against whom You file such action is referred to as "Participant") alleging that:

(a) such Participant’s Contributor Version directly or indirectly infringes any patent, then any and all rights granted by such Participant to You under Sections 2.1 and/or 2.2 of this License shall, upon 60 days notice from Participant terminate prospectively, unless if within 60 days after receipt of notice You either: (i)

agree in writing to pay Participant a mutually agreeable reasonable royalty for Your past and future use of Modifications made by such Participant, or (ii) withdraw Your litigation claim with respect to the Contributor Version against such Participant. If within 60 days of notice, a reasonable royalty and payment arrangement are not mutually agreed upon in writing by the parties or the litigation claim is not withdrawn, the rights granted by Participant to You under Sections 2.1 and/or 2.2 automatically terminate at the expiration of the 60 day notice period specified above.

(b) any software, hardware, or device, other than such Participant's Contributor Version, directly or indirectly infringes any patent, then any rights granted to You by such Participant under Sections 2.1(b) and 2.2(b) are revoked effective as of the date You first made, used, sold, distributed, or had made, Modifications made by that Participant.

8.3. If You assert a patent infringement claim against Participant alleging that such Participant's Contributor Version directly or indirectly infringes any patent where such claim is resolved (such as by license or settlement) prior to the initiation of patent infringement litigation, then the reasonable value of the licenses granted by such Participant under Sections 2.1 or 2.2 shall be taken into account in determining the amount or value of any payment or license.

8.4. In the event of termination under Sections 8.1 or 8.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or any distributor hereunder prior to termination shall survive termination.

9. LIMITATION OF LIABILITY.

UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT (INCLUDING NEGLIGENCE), CONTRACT, OR OTHERWISE, SHALL YOU, THE INITIAL DEVELOPER, ANY OTHER CONTRIBUTOR, OR ANY DISTRIBUTOR OF COVERED CODE, OR ANY SUPPLIER OF ANY OF SUCH PARTIES, BE LIABLE TO ANY PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES, EVEN IF SUCH PARTY SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM SUCH PARTY'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

10. U.S. GOVERNMENT END USERS.

The Covered Code is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire Covered Code with only those

rights set forth herein.

11. MISCELLANEOUS.

This License represents the complete agreement concerning subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This License shall be governed by California law provisions (except to the extent applicable law, if any, provides otherwise), excluding its conflict-of-law provisions. With respect to disputes in which at least one party is a citizen of, or an entity chartered or registered to do business in the United States of America, any litigation relating to this License shall be subject to the jurisdiction of the Federal Courts of the Northern District of California, with venue lying in Santa Clara County, California, with the losing party responsible for costs, including without limitation, court costs and reasonable attorneys' fees and expenses. The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not apply to this License.

12. RESPONSIBILITY FOR CLAIMS.

As between Initial Developer and the Contributors, each party is responsible for claims and damages arising, directly or indirectly, out of its utilization of rights under this License and You agree to work with Initial Developer and Contributors to distribute such responsibility on an equitable basis. Nothing herein is intended or shall be deemed to constitute any admission of liability.

13. MULTIPLE-LICENSED CODE.

Initial Developer may designate portions of the Covered Code as "Multiple-Licensed". "Multiple-Licensed" means that the Initial Developer permits you to utilize portions of the Covered Code under Your choice of the NPL or the alternative licenses, if any, specified by the Initial Developer in the file described in Exhibit A.

EXHIBIT A -Mozilla Public License.

``The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is _____.

The Initial Developer of the Original Code is _____. Portions created by _____ are Copyright (C) _____

```
_____. All Rights Reserved.

Contributor(s): _____.

Alternatively, the contents of this file may be used under the terms
of the _____ license (the  "[___] License"), in which case the
provisions of [______] License are applicable instead of those
above.  If you wish to allow use of your version of this file only
under the terms of the [____] License and not to allow others to use
your version of this file under the MPL, indicate your decision by
deleting  the provisions above and replace  them with the notice and
other provisions required by the [___] License.  If you do not delete
the provisions above, a recipient may use your version of this file
under either the MPL or the [___] License."

[NOTE: The text of this Exhibit A may differ slightly from the text of
the notices in the Source Code files of the Original Code. You should
use the text of this Exhibit A rather than the text found in the
Original Code Source Code for Your Modifications.]
```

14.2.6 Slirp license

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software
   must display the following acknowledgment:
     This product includes software developed by Danny Gasparovski.

THIS SOFTWARE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL
DANNY GASPAROVSKI OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

14.2.7 liblzf license

```
Redistribution and use in source and binary forms, with or without modifica-
tion, are permitted provided that the following conditions are met:

1.  Redistributions of source code must retain the above copyright notice,
    this list of conditions and the following disclaimer.

2.  Redistributions in binary form must reproduce the above copyright
```

notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

14.2.8 GNU General Public License (GPL)

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their

rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1

```
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License.  (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works.  But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of
this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,
```

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent

license would not permit royalty-free redistribution of the Program by
all those who receive copies directly or indirectly through you, then
the only way you could satisfy both it and this License would be to
refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under
any particular circumstance, the balance of the section is intended to
apply and the section as a whole is intended to apply in other
circumstances.

It is not the purpose of this section to induce you to infringe any
patents or other property right claims or to contest validity of any
such claims; this section has the sole purpose of protecting the
integrity of the free software distribution system, which is
implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed
through that system in reliance on consistent application of that
system; it is up to the author/donor to decide if he or she is willing
to distribute software through any other system and a licensee cannot
impose that choice.

This section is intended to make thoroughly clear what is believed to
be a consequence of the rest of this License.

  8. If the distribution and/or use of the Program is restricted in
certain countries either by patents or by copyrighted interfaces, the
original copyright holder who places the Program under this License
may add an explicit geographical distribution limitation excluding
those countries, so that distribution is permitted only in or among
countries not thus excluded. In such case, this License incorporates
the limitation as if written in the body of this License.

  9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time. Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

Each version is given a distinguishing version number. If the Program
specifies a version number of this License which applies to it and "any
later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free
Software Foundation. If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.

  10. If you wish to incorporate parts of the Program into other free
programs whose distribution conditions are different, write to the author
to ask for permission. For software which is copyrighted by the Free
Software Foundation, write to the Free Software Foundation; we sometimes
make exceptions for this. Our decision will be guided by the two goals
of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

     NO WARRANTY

  11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY

FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

14.2.9 lwIP license

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

15 VirtualBox privacy policy

Version 1.1, April 25, 2008

This privacy policy sets out how Sun Microsystems, Inc. ("Sun") treats personal information related to the virtualbox.org website and the VirtualBox registration process.

§ 1 virtualbox.org. The "virtualbox.org" website, as any other website, logs anonymous usage information such as your IP address, geographical location, browser type, referral source, length of visit and number of page views while you visit. In addition, but only if you choose to register, the website's bug tracking and forum services store the data you choose to reveal upon registration, such as your user name and contact information.

§ 2 Cookies. The virtualbox.org website, the bug tracker and the forum services use cookies to identify and track the visiting web browser and, if you have registered, to facilitate login. Most browsers allow you to refuse to accept cookies. While you can still visit the website with cookies disabled, logging into the bug tracker and forum services will most likely not work without them.

§ 3 VirtualBox registration process. VirtualBox asks that the user register with Sun by supplying their name and e-mail address and additionally whether they would or would not like to be notified about product updates by e-mail. Only this information, together with the product version and platform being used, is submitted to Sun and stored together with the IP address of the submitter.

§ 4 Usage of personal information. Sun may use personal data collected by the means above (collectively, "personal data") for statistical purposes as well as to automatically inform you about new notices related to your posts on the bug tracker and forum services, to administer the website and to contact you due to technical issues. Sun will also inform you about new product releases related to VirtualBox, unless you have requested otherwise during registration.

In no event will personal data without your express consent be provided to any third parties, unless Sun may be required to do so by law or in connection with legal proceedings.

§ 5 Updates. Sun may update this privacy policy by posting a new version on the website. You should check this page occasionally to ensure you are happy with any changes.

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 higher as well as Linux 2.4 and higher 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 chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62.

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 ("local APIC"). Many systems also contain an I/O APIC (input output APIC) as a sperate chip which provides more than 16 IRQs. Windows 2000 and higher 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 (synonymous with IDE). See chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62.

C

COM Microsoft Component Object Model, a programming infrastructure for modular software. COM allows applications to provide application programming interfaces which can be accessed from various other programming languages and applications. 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 allows 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. 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*, page 69.

E

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 VirtualBox, we sometimes refer to the main graphical `VirtualBox` program as the "GUI", to differentiate it from the `VBoxManage` interface.

GUID See UUID.

I

IDE Integrated Drive Electronics, an industry standard for hard disk interfaces. See chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62.

I/O APIC See APIC.

iSCSI Internet SCSI; see chapter 5.5, *iSCSI servers*, page 66.

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 as colons, such as `00:17:3A:5E:CB:08`.

N

NAT Network Address Translation. A technique to share networking interfaces by which an interface modifies the source and/or target IP addresses of networking

packages according to specific rules. Commonly employed by routers and firewalls to shield an internal network from the Internet, VirtualBox can use NAT to easily share a host's physical networking hardware with its virtual machines. See chapter 6.4, *Network Address Translation (NAT)*, page 70.

P

PAE Physical Address Extension. This allows accessing more than 4 GB of RAM even in 32-bit environments; see chapter 3.7.1.2, *"Advanced" tab*, page 43.

PIC See APIC.

PXE Preboot Execution Environment, an industry standard for booting PC systems from remote network locations. It includes DHCP for IP configuraiton 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. VirtualBox contains an enhanced implementation of the relevant standards called "VirtualBox RDP" (VRDP), which is largely compatible with Microsoft's RDP implementation. See chapter 7.4, *Remote virtual machines (VRDP support)*, page 90 for details.

S

SATA Serial ATA, an industry standard for hard disk interfaces. See chapter 5.1, *Hard disk controllers: IDE, SATA, AHCI*, page 62.

SCSI Small Computer System Interface. An industry standard for data transfer between devices, especially for storage. See chapter 5.5, *iSCSI servers*, page 66.

SOAP The main protocol that describes messages for webservice operation. Originally, SOAP stood for "Simple Object Access Protocol", but eventually the acronym expansion was dropped, and SOAP is now supposed to just stand for itself. Together with WSDL, SOAP is the most important webservice standard. Two versions are in common use today, 1.1 and 1.2. See chapter 10.1.7.1, *SOAP messages*, page 135 for details.

U

UUID A Universally Unique Identifier – often also called GUID (Globally Unique Identifier) – is a string of numbers and letters which can be computed dynamically and is guaranteed to be unique. Generally, it used as a global handle to identify entities. VirtualBox makes use of UUIDs to identify VMs, Virtual Disk Images (VDI files) and other entities.

V

VM Virtual Machine – a virtual computer that VirtualBox allows you to run on top of your actual hardware. See chapter 1.1, *Virtualization basics*, page 8 for details.

VRDP See RDP.

W

WSDL The Web Services Description Language, an XML dialect that describes what operations a webservice offers. See chapter 10.1.7.2, *Service descriptions in WSDL*, page 136 for details.

X

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 allows applications to provide a modular programming interface. VirtualBox makes use of XPCOM on Linux both internally and externally to provide a comprehensive API to third-party developers.