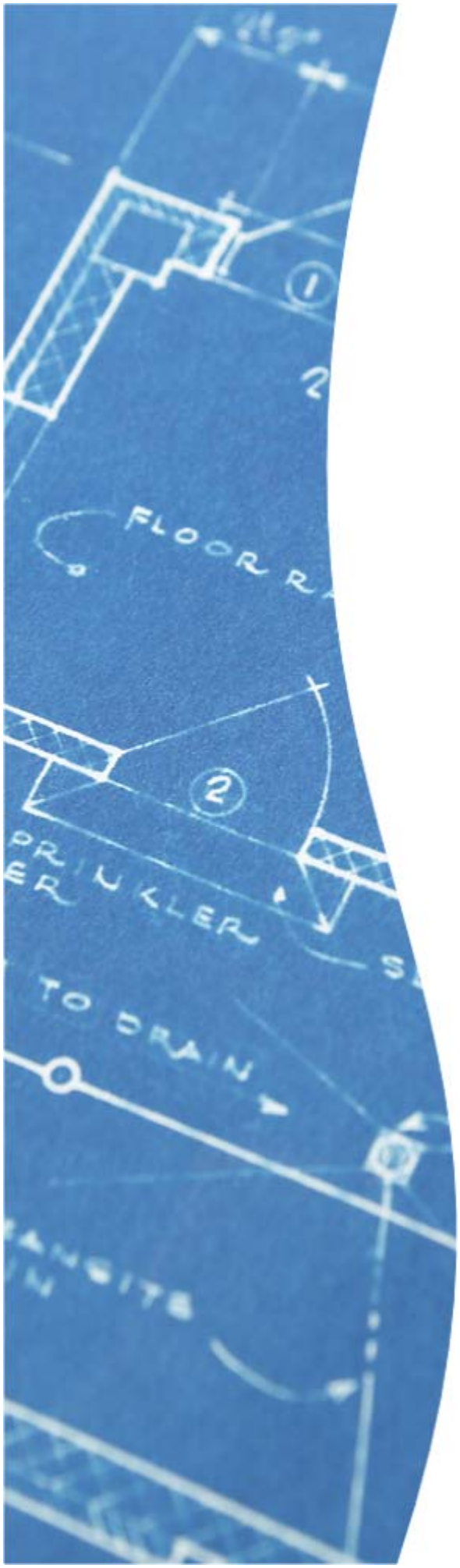

OPTIMIZING THE DESKTOP USING SUN™ XVM VIRTUALBOX

Ulrich Möller, VirtualBox Software Team

Sun BluePrints™ Online

Part No 820-7121-10
Revision 1.0, 11/25/08

Table of Contents

Optimizing the Desktop Using Sun™ xVM VirtualBox

Often users must access applications in operating system (OS) environments other than the native OS on their personal laptop or desktop system. Companies that undertake migration initiatives to open source desktop platforms, for example, sometimes need to provide users with access to critical legacy applications. In other cases, when a user desktop platform is upgraded to state-of-the-art hardware, older operating environments may be no longer supported as the native OS on the new hardware platform. In these cases, desktop virtualization can be a cost-effective answer. Sun xVM VirtualBox is an especially compelling desktop virtualization solution since it is available as a no-cost download for personal use and as open source code. For OEMs and product developers, VirtualBox also provides a comprehensive set of development tools to integrate customized functionality. Some developers use these tools to build a complete, patched, virtualized OS instance and simplify deployment of a precise execution environment for their device and application.

This article introduces Sun xVM VirtualBox, highlighting key product features and benefits. It describes how the software enables multiple virtual machines (VMs) on a single desktop, allowing users to access other operating systems within a native OS environment. The article addresses the following topics:

- "Sun xVM VirtualBox and Desktop Virtualization" on page 1 describes the product and its desktop virtualization capabilities.
- "Flexible Deployment Options" on page 6 explains how the software's flexibility aids in common deployment scenarios.
- "VirtualBox Architecture and Tools" on page 8 discusses the software's modular architecture and tools available to help developers craft customized solutions.
- "Creating a Virtual Machine on a Remote Server" on page 10 steps through a basic VM set-up procedure, showing how easy it is to configure a VM on a remote RDP server using the Command Line Interface (CLI).

This article is intended for a wide audience that includes general users, developers, and third-party OEM vendors. It assumes that users have limited or no previous experience with desktop virtualization. It also describes advanced functionality and distinctive product capabilities that may be of interest to developers and system administrators interested in embedding virtualization technology or deploying more complex virtualization solutions.

Sun xVM VirtualBox and Desktop Virtualization

The concept of virtualization is familiar to most businesses as a means of consolidating workloads and reducing costs. Server virtualization is recognized as a cost-effective way to condense enterprise application services on a single server, which can help to

increase server utilization, improve manageability, reduce power use, and conserve datacenter footprint. Since virtualization technology has evolved in recent years to allow workloads to be securely partitioned on a single server, multiple application tiers or many OS environments can now be easily consolidated on a single physical machine.

Conceptually similar to server virtualization, desktop virtualization allows individual users to access strategic applications on multiple operating systems on a laptop or desktop, with similar benefits that extend from consolidating many virtual machines on a single virtualized platform. The native OS environment on the desktop is called the *host* environment, while operating system instances within virtual machines are called *guest* operating systems. As an example, Figure 1 shows a virtualized desktop with application windows that allow a user to run applications on Windows XP, Ubuntu Linux, and the OpenSolaris™ Operating System (OS) in addition to the native host Mac OS.

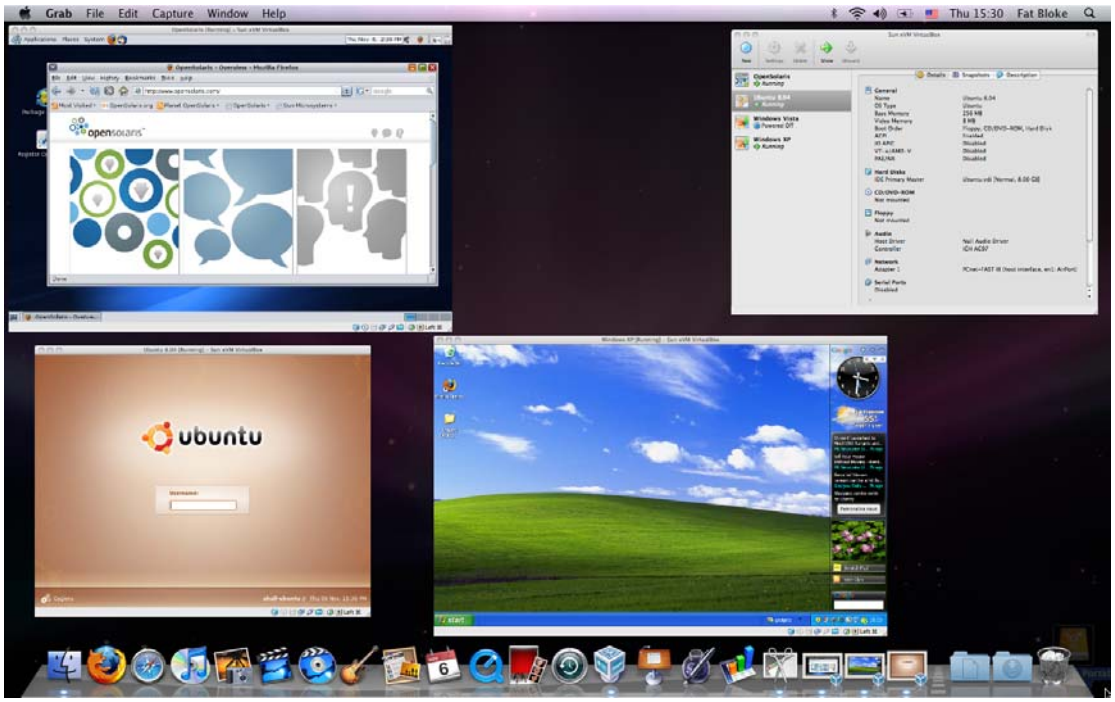

Figure 1. Multiple guest operating environments run on a single physical machine.

With over 7.8 million downloads to date, Sun xVM VirtualBox can be downloaded at no-cost for personal use from http://www.sun.com/virtualbox. Sun is encouraging mass adoption of the software in this respect, building momentum for the product and for desktop virtualization as a whole.

For enterprise deployments, Sun offers an extremely cost-competitive subscription service that includes global 24/7 telephone support. Sun also supplies basic product functionality in the VirtualBox Open Source Edition (OSE) under GPLv2 licensing. The software's availability as open source helps to increase mind-share and encourages an active development community. Sun's philosophy is that open source nurtures innovation and developer momentum, which helps to promote standards and interoperability, which is ultimately to customer advantage.

Unlike many competitive desktop virtualization products, Sun xVM VirtualBox provides broad cross-platform support. VirtualBox binaries are available across many x86 host operating systems — Windows, Mac OS, Linux, the Solaris™ OS, and OpenSolaris™ environments — even on 64-bit as well as 32-bit OS platforms. VirtualBox can host the execution of practically any x86-based guest environment at near-native speeds, and offers powerful virtual hardware and device support. Other competitive products are limited in that they often require separate products specific to a particular host (e.g., one product for Mac OS hosts versus a different product altogether for Windows or Linux hosts). With VirtualBox, the same product functionality is deployed across a range of enterprise desktop host platforms.

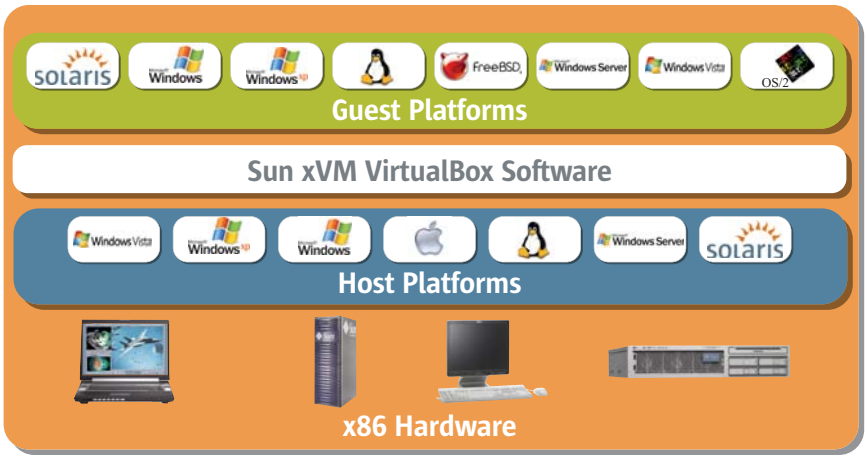

Figure 2. VirtualBox offers broad cross-platform support, with a single product that runs on many hosts and supports many guest OS environments.

Easy-to-use GUI and Comprehensive Tools

The software comes with an easy-to-use, wizard-like GUI that helps users build, launch, and manage created VMs. Beyond the product's simple download-and-go capabilities, VirtualBox includes a comprehensive toolbox that includes a Command Line Interface (CLI), Application Programming Interfaces (APIs), and a Software Development Kit (SDK). Described in more detail later in this article, the product's modular architecture, along with the CLI, APIs, and SDK, are useful in crafting customized solutions. The tools are particularly helpful to OEM developers because they can encapsulate a precise OS

environment into a VM (including patches, application libraries, and a target OS). The developer can then distribute the VM package to their customers, who can activate it to deploy a pre-determined and comprehensive execution environment.

Usability Features

The software includes a number of features that enrich its usability:

- *Near-native performance.* The software ships with optimizations (called "guest additions") for certain guest operating systems. Guest additions are installed inside a specific VM to accelerate performance or otherwise enhance the feature set of a particular guest environment.
- *Rich host and guest integration.* When a VM is configured in "seamless" mode, the software maps guest application windows directly on the host desktop, allowing guest and host windows to appear side-by-side. This gives the impression that a guest application window is running natively. Just as with native OS windows, the user can dynamically resize guest application windows as needed, or alternatively, run the guest VM in full-screen mode.
- *Shared folders and shared clipboard.* Applications running on host and guest operating systems can transparently share files and clipboard data. When a folder is configured as a shared folder, it becomes available to the guest environment as a network share. Optionally, the clipboard of a guest OS can be shared with the host OS.
- *Time synchronization.* Since the native OS manages the host system's clock, some virtualized desktops do not adequately address time synchronization between host and guest OS environments. With VirtualBox, the guest operating environment is synchronized automatically with the native OS, allowing the user to see accurate time reporting.
- *Powerful virtualized networking.* Up to four virtual Gigabit-based network interfaces are supported in each VM and are then accessible from other machines on the physical network. When a VM is configured, the user individually selects what kind of hardware is presented to the virtual machine. Virtual network interfaces also support remote booting via PXE protocol.
- *Robust virtualized hard disk controller support.* VirtualBox implements virtualized instances of two of the most common hard disk controller types, IDE and SATA. Up to 32 SATA devices are supported in a guest OS, enabling access to a variety of device types, including devices with removable media.
- *Support for Advanced Configuration and Power Interface (ACPI) reporting.* This functionality allows an ACPI-aware guest OS to obtain host configuration specifics, which helps the guest configure virtual hardware capabilities. ACPI simplifies the

task of cloning PC images from real machines or from third-party virtual machines into VirtualBox software. An ACPI-aware guest can also obtain host power status, such as a low battery condition on a laptop.

- *Virtualized display support.* VirtualBox supports all VESA standard resolutions and color depths. VMs can also be configured to support screen resolutions many times that of a typical physical display. This allows VMs to be spread over a large number of physical screens on a host system.
- *Secure VM instances.* Because VirtualBox prevents data leakage, VM instances are isolated and cannot impact applications running on the host OS or within other VMs. This allows companies to extend access to non-native applications while protecting core business functions.

Advanced Features

VirtualBox abstracts each virtual machine and its associated disk storage into individual VM containers — a design that enables distinctive and advanced functionality:

- *Snapshot/rollback.* VirtualBox can save point-in-time backups of VM states. (This capability is similar to Time Machine's ability to snapshot the OS environment in Mac OS.) The snapshot/rollback functionality is powerful because it allows a user to revert easily to a VM's known good state. This capability also helps to address disaster recovery requirements since a VM can be restored quickly in the event of malfunctioning software, virus contamination, or system loss. VM snapshots can be woken up, copied, and transported between hosts, so they offer tremendous flexibility to administrators and OEM developers who must manage and quickly deploy VM environments.
- *iSCSI storage capability.* VirtualBox has built-in iSCSI support that allows VMs to be housed and transparently accessed on remote iSCSI storage. In departmental or enterprise deployments, centralized and remote iSCSI storage can greatly simplify VM management.
- *Remote access through RDP.* VirtualBox enables remote VM access through an RDP server. This allows a user to run a VM on a remote server and display it locally through RDP protocol. These built-in RDP capabilities allow a remote system administrator, for example, to access an enterprise desktop and troubleshoot a locally running VM.
- *Support for USB peripherals.* The software includes strong support for USB-connected peripheral devices, such as USB fingerprint readers, printers, memory sticks, etc. Even over RDP, VMs that are executing remotely can be configured to enable access to locally connected USB devices.

Flexible Deployment Options

Because of these advanced features and the product's modular design, Sun xVM VirtualBox is extremely flexible and allows an almost infinite range of deployment possibilities. Figure 3 shows four typical usage scenarios. The left side of Figure 3 shows the most common use case, which is simply when a user concurrently accesses multiple OS environments on a personal laptop or desktop. As shown, the software runs locally (using local VM storage), creating a virtualized multiplatform desktop.

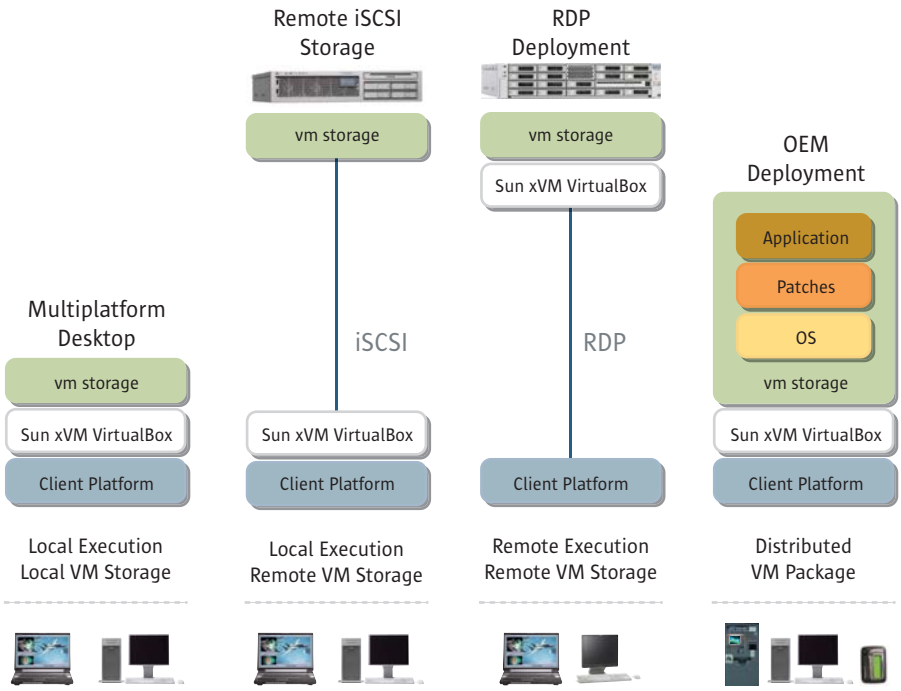

Figure 3. Many deployment scenarios are possible because of the software's flexibility and modular design.

A typical user in this case is a software developer who downloads VirtualBox (at no charge for personal use), installs it on a desktop, and creates several VMs to simulate test environments for potential deployments. On a virtualized platform, multiple VMs act as isolated sandboxes for testing, eliminating the need and cost associated with corresponding physical systems. For example, a Web 2.0 developer must test applications on multiple browsers including Safari, Internet Explorer, and Mozilla (commonly used in Mac OS, Microsoft Windows, and UNIX environments, respectively). A single laptop configured with multiple VMs and these operating systems becomes a powerful yet low-cost test and development environment.

Often a number of enterprise users require virtualized desktops. Organizations who transition to open source desktops sometimes still need to access proprietary OS environments, and some companies have successfully deployed VirtualBox to meet this

need. In the enterprise VirtualBox can be configured to enable access to critical legacy applications, reducing training time and conserving valuable IT resources. The software's snapshot capabilities greatly simplify deployment since VMs can be easily captured and duplicated across desktop systems. Sun offers an extremely cost-competitive enterprise subscription service that includes global 24/7 support.

Enterprise deployments can also take advantage of remote iSCSI storage and RDP capabilities built into VirtualBox (Figure 3). Using iSCSI storage, VMs can be centralized and stored remotely to simplify administrative tasks. Executing locally, VirtualBox can access remote VMs on iSCSI storage. RDP capabilities also allow the software to execute on a remote server, displaying VMs on the local desktop (for sample steps to do this, see "Creating a Virtual Machine on a Remote Server" on page 10). Even when executing on a remote RDP server, the software can be configured to access locally connected USB peripheral devices.

Enterprise deployments can benefit from the security offered by compartmentalized VM instances. Guest OS instances run in isolation from the host OS instance, and the software is designed to prevent data leakage. Virtualized desktops allow businesses to grant users access to non-native applications while protecting mission-critical business applications.

VirtualBox features well-defined internal programming interfaces, a Software Developer's Kit, and an extensive Command Line Interface — the tools necessary to integrate customized functionality and interface with the software programmatically. The software's small footprint (approximately 20MB) makes it an attractive building block for OEM and other developers. Some OEMs take advantage of the software's VM snapshot capabilities to simplify the distribution of an OS stack (right side, Figure 3). A self-contained VM with a patched OS, required application libraries, device drivers, and corresponding application software creates a comprehensive, easy-to-install, and precisely defined execution environment.

VirtualBox Architecture and Tools

VirtualBox features a layered, extensible, modular design that includes comprehensive development tools and interfaces (Figure 4). A hypervisor layer — the core of the virtualization engine that controls VM execution — underlies the architectural stack.

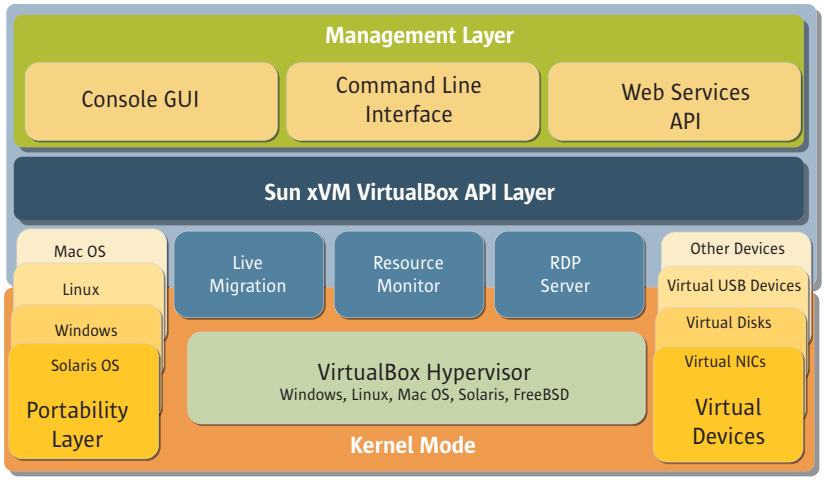

Figure 4. A modular architecture facilitates the ability to integrate customized third party functionality.

Management Layer

VirtualBox GUI Console

When VirtualBox is installed on a desktop platform, a user typically brings up the built-in graphical user interface (GUI) to install, configure, and manage multiple virtual machines on the desktop (Figure 5). Equivalent management functionality is provided through the `VBoxManage` Command Line Interface (CLI) as well as through an API. The CLI and API interfaces allow developers to integrate VirtualBox into third-party products and to control the software and manage virtual machines programmatically.

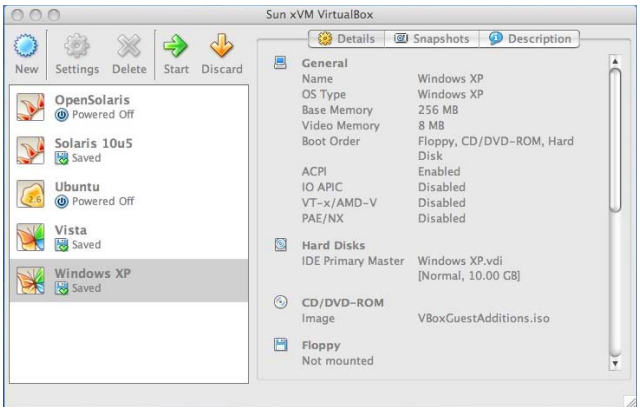

Figure 5. The built-in GUI simplifies VM creation and management for users.

VirtualBox CLI

At the command line, entering `VBoxManage` without any parameters lists the entire CLI syntax, including all available parameters. `VBoxManage` commands can create, configure, modify, start, stop, and capture VMs via snapshot — all of the same functionality that can be performed via the GUI. The Sun xVM VirtualBox User's Manual (available for download with the software or on http://virtualbox.org) lists the syntax of `VBoxManage` CLI commands.

VirtualBox API Layer

Referring back to Figure 4, the VirtualBox API layer provides the underlying foundation for the management layer. This API, also known as the VirtualBox Main API, exposes the entire feature set of the virtualization engine. It is completely documented and can be used to control VirtualBox programmatically.

With the VirtualBox Main API, an application can create, configure, start, stop, and delete virtual machine instances. It is also possible to retrieve performance statistics about running VMs. In fact, the provided management front-ends — the GUI console and the `VBoxManage` CLI — are based on the Main API.

There are primarily two ways in which the Main API can be called by other code: either through the provided Web Service or through the Component Object Model (COM).

VirtualBox includes a Web Service capability that maps almost all of the API. It ships as a stand-alone executable (`vboxwebsrv`) that acts as an HTTP server, accepts SOAP (Simple Object Access Protocol) connections, and processes them either remotely or locally. Since the Web Service is publicly described in Web Service Description File (WSDL) format, client programs can call it using any language with a toolkit that understands WSDL. Such toolkits are available for most programming languages (e.g., for Java, C++, .NET, PHP, Python, Perl, etc.).

As an alternative to using the Web Service shipped with the VirtualBox software distribution, a developer familiar with Python or C++ can access the Main API via the Component Object Model (COM). COM is an interprocess mechanism for software components originally introduced by Microsoft for Microsoft Windows. Using 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 can communicate with it. COM also cleanly separates interface from implementation, so external programs need not know anything about the details of VirtualBox internals.

On a Windows host, VirtualBox uses COM functionality that is native to Windows operating systems. On other hosts (including Linux), VirtualBox comes with a built-in implementation of XPCOM (Cross Platform Component Object Model, originally created

by the Mozilla project) which is enhanced to support interprocess communication. Internally VirtualBox has an abstraction layer that allows the same user code to work with Microsoft COM as well as with the XPCOM implementation.

As part of its comprehensive support for third-party developers, VirtualBox offers a Software Developer's Kit (SDK). The SDK contains all the documentation and interface files needed to write code that interacts with VirtualBox, along with code examples. The SDK is available as a no-charge download at http://www.sun.com/virtualbox. It also includes the *Sun xVM VirtualBox Programming Guide and Reference Manual*, which describes programmatic control of VirtualBox using both Web Service and COM applications that access the Main API.

Creating a Virtual Machine on a Remote Server

The example procedure below constructs a virtual machine on a remote headless server using RDP capabilities built into the VirtualBox software. The procedure shows how to use Command Line Interface commands to create a virtual machine, establish an RDP connection, and install a guest operating system. The steps are performed remotely from a client desktop without a need to physically touch the server.

The procedure assumes the following:

- VirtualBox is already installed on the remote server on one of the supported host operating systems. (The steps given below were used to create a VM on a Linux host.)
- An ISO file exists on the server that contains the installation image of the guest operating system (this example creates a VM using Windows XP as the guest OS).
- A terminal connection exists to the remote server and can be accessed via the command line (e.g., via `telnet` or `ssh`).
- An RDP viewer exists on the client desktop such as `rdesktop` on a Linux client, or the RDP viewer on a Windows machine (the RDP viewer is usually found in "Accessories" -> "Communication" -> "Remote Desktop Connection").

The following steps set up a virtual machine on the remote headless server:

1. After logging into the remote server (via `telnet` or `ssh`), create a new virtual machine and register the VM instance within the VirtualBox software:

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

Note that if you do not specify the "`-register`" argument then you will have to manually use the `registervm` command later. Registering the VM allows VM definitions to be duplicated on other machines and then imported and registered.

2. Make sure the settings for the VM are appropriate for the guest operating system to be installed. For example:

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

The "`-memory`" argument establishes the amount of physical RAM (in MB) that the virtual machine should allocate for itself from the host. The argument "`-acpi on`" indicates that the guest OS can subsequently take advantage of Advanced Configuration and Power Interface (ACPI) reporting. The "`-boot1 dvd`" argument tells the VM to attempt to use a local DVD device as the first boot device. Lastly, the argument "`-nic1 nat`" defines what type of networking should be available to the virtual machine.

3. Next, 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 image to be installed later:

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

6. Attach the ISO image to the virtual machine, allowing the VM to boot from the attached image:

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

Alternatively, the following command inserts a DVD image into the virtual machine from the host's DVD drive, without having to register the image first:

```
# VBoxManage controlvm dvdattach
```

7. Once the operating system image is specified to the VM, then start the virtual machine:

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

The command `VBoxHeadless` is simply an alternate front-end to start the VM instance without producing any visible output on the host. After executing this command, the copyright notice should appear. If instead the command line returns, then something is wrong (e.g., the VM was not properly configured or insufficient host resources exist).

8. On the client machine, initialize the RDP connection (using either `rdesktop` or the RDP viewer) and connect to the server. Assuming a Linux client with `rdesktop`, enter the following:

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

The "`-a 16`" option of requests a color depth of 16 bits per pixel, which is recommended as the default. After the installation, set the color depth of the guest operating system to the same value. At this point in the installation process, messages from the guest OS are now visible via the `rdesktop` connection.

The process above allows VirtualBox to execute on the remote server while displaying resulting VM windows on the desktop client. This can be useful, for example, if a user knows the remote server instance and can specifically access that server to run the VirtualBox software. This capability is also helpful when an administrator wants to troubleshoot a remote VirtualBox installation and display the results on a local desktop.

Summary

Sun xVM VirtualBox offers extensive flexibility to meet various desktop virtualization needs for a spectrum of users. Individual users can download VirtualBox at no charge for personal use (or download the open source) to create a powerful multiplatform desktop where almost any x86 operating system — including OpenSolaris, the Solaris OS, Linux, Windows, and Mac OS — can coexist. For example, VirtualBox provides a compelling test and development environment for application developers and eliminates the expense and management overhead of additional physical platforms. With competitive license subscriptions now available, enterprise customers can also download, distribute, and install VirtualBox software across the business, backed up by 24/7 hotline support from Sun. Because VirtualBox features snapshot capabilities, remote iSCSI, and RDP support, and is a singe product that supports many host and guest combinations, it offers some distinctive functionality over competitive desktop virtualization products.

VirtualBox also features a comprehensive toolbox that includes CLI and API capabilities. These tools make it easy for innovative developers to embed virtualization functionality into third-party products. Developers can create alternate front-ends for the VirtualBox, or programmatically create and manage virtual machines. To deploy a precise execution environment, some manufacturers are using VirtualBox to distribute a VM

snapshot containing a patched OS, application libraries, and required device drivers. VirtualBox provides the tool set and the flexibility needed so that developers can easily integrate virtualization's potential into their own products.

About the Author

Ulrich Möller is both a lawyer and developer. He was part of the VirtualBox team that was acquired by Sun Microsystems in February 2008, holds a Ph.D. in copyright law from the Humboldt University in Berlin and has executed much of the original open sourcing of VirtualBox back in 2007. At the same time, he has been part of the VirtualBox developer team for years and has written most of the developer tools and documentation as well as the *Sun xVM VirtualBox User Manual*. He has been living in Berlin, Germany since 1996.

References

Sun xVM VirtualBox User Manual, Sun Microsystems, Inc., 2008.

Sun xVM VirtualBox Programming Guide and Reference, Sun Microsystems, Inc., 2008.

Sun xVM VirtualBox software: `http://www.sun.com/virtualbox`

xVM VirtualBox open source software and community: `http://virtualbox.org`

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is
`http://docs.sun.com/`

To reference Sun BluePrints Online articles, visit the Sun BluePrints Online Web site at:
`http://www.sun.com/blueprints/online.html`

Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 USA **Phone** 1-650-960-1300 or 1-800-555-9SUN (9786) **Web** sun.com

Printed in USA 11/08