Oracle VM VirtualBox®

User Manual

Version 5.1.22

© 2004-2017 Oracle Corporation

http://www.virtualbox.org

Contents

1 First steps **11**
1.1 Why is virtualization useful? . . . 12
1.2 Some terminology . . . 12
1.3 Features overview . . . 13
1.4 Supported host operating systems . . . 15
1.5 Installing VirtualBox and extension packs . . . 16
1.6 Starting VirtualBox . . . 17
1.7 Creating your first virtual machine . . . 18
1.8 Running your virtual machine . . . 21
1.8.1 Starting a new VM for the first time . . . 21
1.8.2 Capturing and releasing keyboard and mouse . . . 22
1.8.3 Typing special characters . . . 23
1.8.4 Changing removable media . . . 24
1.8.5 Resizing the machine's window . . . 24
1.8.6 Saving the state of the machine . . . 25
1.9 Using VM groups . . . 26
1.10 Snapshots . . . 26
1.10.1 Taking, restoring and deleting snapshots . . . 27
1.10.2 Snapshot contents . . . 28
1.11 Virtual machine configuration . . . 29
1.12 Removing virtual machines . . . 30
1.13 Cloning virtual machines . . . 30
1.14 Importing and exporting virtual machines . . . 31
1.15 Global Settings . . . 32
1.16 Alternative front-ends . . . 32

2 Installation details **34**
2.1 Installing on Windows hosts . . . 34
2.1.1 Prerequisites . . . 34
2.1.2 Performing the installation . . . 34
2.1.3 Uninstallation . . . 36
2.1.4 Unattended installation . . . 36
2.1.5 Public properties . . . 36
2.2 Installing on Mac OS X hosts . . . 36
2.2.1 Performing the installation . . . 36
2.2.2 Uninstallation . . . 37
2.2.3 Unattended installation . . . 37
2.3 Installing on Linux hosts . . . 37
2.3.1 Prerequisites . . . 37
2.3.2 The VirtualBox driver modules . . . 37
2.3.3 Performing the installation . . . 38
2.3.4 The vboxusers group . . . 41
2.3.5 Starting VirtualBox on Linux . . . 42
2.4 Installing on Solaris hosts . . . 42
2.4.1 Performing the installation . . . 42
2.4.2 The vboxuser group . . . 43

2.4.3 Starting VirtualBox on Solaris . . . 43
2.4.4 Uninstallation . . . 43
2.4.5 Unattended installation . . . 43
2.4.6 Configuring a zone for running VirtualBox . . . 43

3 Configuring virtual machines **45**
3.1 Supported guest operating systems . . . 45
3.1.1 Mac OS X guests . . . 46
3.1.2 64-bit guests . . . 46
3.2 Emulated hardware . . . 47
3.3 General settings . . . 48
3.3.1 “Basic” tab . . . 48
3.3.2 “Advanced” tab . . . 48
3.3.3 “Description” tab . . . 49
3.4 System settings . . . 49
3.4.1 “Motherboard” tab . . . 49
3.4.2 “Processor” tab . . . 51
3.4.3 “Acceleration” tab . . . 51
3.5 Display settings . . . 51
3.6 Storage settings . . . 52
3.7 Audio settings . . . 54
3.8 Network settings . . . 55
3.9 Serial ports . . . 55
3.10 USB support . . . 56
3.10.1 USB settings . . . 56
3.10.2 Implementation notes for Windows and Linux hosts . . . 58
3.11 Shared folders . . . 58
3.12 User Interface . . . 58
3.13 Alternative firmware (EFI) . . . 59
3.13.1 Video modes in EFI . . . 59
3.13.2 Specifying boot arguments . . . 59

4 Guest Additions **60**
4.1 Introduction . . . 60
4.2 Installing and Maintaining Guest Additions . . . 61
4.2.1 Guest Additions for Windows . . . 61
4.2.2 Guest Additions for Linux . . . 65
4.2.3 Guest Additions for Solaris . . . 67
4.2.4 Guest Additions for OS/2 . . . 67
4.3 Shared folders . . . 68
4.3.1 Manual mounting . . . 68
4.3.2 Automatic mounting . . . 69
4.4 Drag and Drop . . . 70
4.4.1 Supported formats . . . 71
4.4.2 Known limitations . . . 72
4.5 Hardware-accelerated graphics . . . 72
4.5.1 Hardware 3D acceleration (OpenGL and Direct3D 8/9) . . . 72
4.5.2 Hardware 2D video acceleration for Windows guests . . . 73
4.6 Seamless windows . . . 73
4.7 Guest properties . . . 74
4.8 Guest control . . . 76
4.9 Memory overcommitment . . . 76
4.9.1 Memory ballooning . . . 76
4.9.2 Page Fusion . . . 77

5 Virtual storage **79**
5.1 Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe 79
5.2 Disk image files (VDI, VMDK, VHD, HDD) 82
5.3 The Virtual Media Manager 83
5.4 Special image write modes 84
5.5 Differencing images 86
5.6 Cloning disk images 88
5.7 Host I/O caching 89
5.8 Limiting bandwidth for disk images 89
5.9 CD/DVD support 90
5.10 iSCSI servers 91

6 Virtual networking **92**
6.1 Virtual networking hardware 92
6.2 Introduction to networking modes 93
6.3 Network Address Translation (NAT) 94
6.3.1 Configuring port forwarding with NAT 94
6.3.2 PXE booting with NAT 95
6.3.3 NAT limitations 95
6.4 Network Address Translation Service 96
6.5 Bridged networking 97
6.6 Internal networking 98
6.7 Host-only networking 99
6.8 UDP Tunnel networking 100
6.9 VDE networking 101
6.10 Limiting bandwidth for network I/O 101
6.11 Improving network performance 102

7 Remote virtual machines **104**
7.1 Remote display (VRDP support) 104
7.1.1 Common third-party RDP viewers 104
7.1.2 VBoxHeadless, the remote desktop server 105
7.1.3 Step by step: creating a virtual machine on a headless server 106
7.1.4 Remote USB 108
7.1.5 RDP authentication 108
7.1.6 RDP encryption 109
7.1.7 Multiple connections to the VRDP server 110
7.1.8 Multiple remote monitors 111
7.1.9 VRDP video redirection 111
7.1.10 VRDP customization 111
7.2 Teleporting 112

8 VBoxManage **114**
8.1 Introduction 114
8.2 Commands overview 115
8.3 General options 125
8.4 VBoxManage list 125
8.5 VBoxManage showvminfo 126
8.6 VBoxManage registervm / unregistervm 127
8.7 VBoxManage createvm 127
8.8 VBoxManage modifyvm 128
8.8.1 General settings 128
8.8.2 Networking settings 131
8.8.3 Miscellaneous settings 134

8.8.4 Video Capture settings . . . 135
8.8.5 Remote machine settings . . . 136
8.8.6 Teleporting settings . . . 139
8.8.7 Debugging settings . . . 139
8.8.8 USB card reader settings . . . 139
8.8.9 Auto starting VMs during host system boot . . . 140
8.9 VBoxManage clonevm . . . 140
8.10 VBoxManage import . . . 141
8.11 VBoxManage export . . . 142
8.12 VBoxManage startvm . . . 142
8.13 VBoxManage controlvm . . . 143
8.14 VBoxManage discardstate . . . 149
8.15 VBoxManage adoptstate . . . 149
8.16 VBoxManage snapshot . . . 149
8.17 VBoxManage closemedium . . . 150
8.18 VBoxManage storageattach . . . 150
8.19 VBoxManage storagectl . . . 154
8.20 VBoxManage bandwidthctl . . . 154
8.21 VBoxManage showmediuminfo . . . 155
8.22 VBoxManage createhd . . . 155
8.23 VBoxManage modifymedium . . . 156
8.24 VBoxManage clonemedium . . . 157
8.25 VBoxManage mediumproperty . . . 158
8.26 VBoxManage encryptmedium . . . 159
8.27 VBoxManage checkmediumpwd . . . 159
8.28 VBoxManage convertfromraw . . . 160
8.29 VBoxManage getextradata/setextradata . . . 160
8.30 VBoxManage setproperty . . . 161
8.31 VBoxManage usbfilter add/modify/remove . . . 161
8.32 VBoxManage sharedfolder add/remove . . . 163
8.33 VBoxManage guestproperty . . . 164
8.34 VBoxManage guestcontrol . . . 165
8.35 VBoxManage metrics . . . 172
8.36 VBoxManage natnetwork . . . 174
8.37 VBoxManage hostonlyif . . . 176
8.38 VBoxManage dhcpserver . . . 176
8.39 VBoxManage usbdevsource . . . 177
8.40 VBoxManage debugvm . . . 177
8.41 VBoxManage extpack . . . 183

9 Advanced topics **185**
9.1 VBoxSDL, the simplified VM displayer . . . 185
9.1.1 Introduction . . . 185
9.1.2 Secure labeling with VBoxSDL . . . 185
9.1.3 Releasing modifiers with VBoxSDL on Linux . . . 186
9.2 Automated guest logons . . . 187
9.2.1 Automated Windows guest logons . . . 187
9.2.2 Automated Linux/Unix guest logons . . . 188
9.3 Advanced configuration for Windows guests . . . 191
9.3.1 Automated Windows system preparation . . . 191
9.4 Advanced configuration for Linux and Solaris guests . . . 192
9.4.1 Manual setup of selected guest services on Linux . . . 192
9.4.2 Guest graphics and mouse driver setup in depth . . . 192
9.5 CPU hot-plugging . . . 193

9.6 PCI passthrough . . . 194
9.7 Webcam passthrough . . . 195
9.7.1 Using a host webcam in the guest . . . 195
9.7.2 Windows hosts . . . 196
9.7.3 Mac OS X hosts . . . 196
9.7.4 Linux and Solaris hosts . . . 197
9.8 Advanced display configuration . . . 197
9.8.1 Custom VESA resolutions . . . 197
9.8.2 Configuring the maximum resolution of guests when using the graphical frontend . . . 197
9.9 Advanced storage configuration . . . 198
9.9.1 Using a raw host hard disk from a guest . . . 198
9.9.2 Configuring the hard disk vendor product data (VPD) . . . 200
9.9.3 Access iSCSI targets via Internal Networking . . . 200
9.10 Legacy commands for using serial ports . . . 201
9.11 Fine-tuning the VirtualBox NAT engine . . . 201
9.11.1 Configuring the address of a NAT network interface . . . 201
9.11.2 Configuring the boot server (next server) of a NAT network interface . . . 202
9.11.3 Tuning TCP/IP buffers for NAT . . . 202
9.11.4 Binding NAT sockets to a specific interface . . . 202
9.11.5 Enabling DNS proxy in NAT mode . . . 202
9.11.6 Using the host's resolver as a DNS proxy in NAT mode . . . 203
9.11.7 Configuring aliasing of the NAT engine . . . 203
9.12 Configuring the BIOS DMI information . . . 204
9.13 Configuring the custom ACPI table . . . 205
9.14 Fine-tuning timers and time synchronization . . . 206
9.14.1 Configuring the guest time stamp counter (TSC) to reflect guest execution . . . 206
9.14.2 Accelerate or slow down the guest clock . . . 206
9.14.3 Tuning the Guest Additions time synchronization parameters . . . 206
9.14.4 Disabling the Guest Additions time synchronization . . . 207
9.15 Installing the alternate bridged networking driver on Solaris 11 hosts . . . 207
9.16 VirtualBox VNIC templates for VLANs on Solaris 11 hosts . . . 208
9.17 Configuring multiple host-only network interfaces on Solaris hosts . . . 208
9.18 Configuring the VirtualBox CoreDumper on Solaris hosts . . . 209
9.19 VirtualBox and Solaris kernel zones . . . 210
9.20 Locking down the VirtualBox GUI . . . 210
9.20.1 Customizing the VM manager . . . 210
9.20.2 VM selector customization . . . 211
9.20.3 Configure VM selector menu entries . . . 211
9.20.4 Configure VM window menu entries . . . 212
9.20.5 Configure VM window status bar entries . . . 215
9.20.6 Configure VM window visual modes . . . 215
9.20.7 Host Key customization . . . 216
9.20.8 Action when terminating the VM . . . 216
9.20.9 Default action when terminating the VM . . . 217
9.20.10 Action for handling a Guru Meditation . . . 217
9.20.11 Configuring automatic mouse capturing . . . 217
9.20.12 Configuring automatic mouse capturing . . . 218
9.20.13 Requesting legacy full-screen mode . . . 218
9.21 Starting the VirtualBox web service automatically . . . 218
9.21.1 Linux: starting the webservice via `init` . . . 218
9.21.2 Solaris: starting the web service via SMF . . . 219

9.21.3 Mac OS X: starting the webservice via launchd . . . 219
9.22 VirtualBox Watchdog . . . 220
9.22.1 Memory ballooning control . . . 220
9.22.2 Host isolation detection . . . 221
9.22.3 More information . . . 222
9.22.4 Linux: starting the watchdog service via `init` . . . 222
9.22.5 Solaris: starting the watchdog service via SMF . . . 222
9.23 Other extension packs . . . 223
9.24 Starting virtual machines during system boot . . . 223
9.24.1 Linux: starting the autostart service via `init` . . . 223
9.24.2 Solaris: starting the autostart service via SMF . . . 224
9.24.3 Mac OS X: starting the autostart service via launchd . . . 224
9.25 VirtualBox expert storage management . . . 224
9.26 Handling of host power management events . . . 225
9.27 Experimental support for passing through SSE4.1 / SSE4.2 instructions . . . 225
9.28 Support for keyboard indicators synchronization . . . 225
9.29 Capturing USB traffic for selected devices . . . 226
9.30 Configuring the heartbeat service . . . 226
9.31 Encryption of disk images . . . 226
9.31.1 Limitations . . . 227
9.31.2 Encrypting disk images . . . 227
9.31.3 Starting a VM with encrypted images . . . 228
9.31.4 Decrypting encrypted images . . . 228
9.32 Paravirtualized debugging . . . 228
9.32.1 Hyper-V debug options . . . 228
9.33 PC speaker passthrough . . . 231
9.34 Accessing USB devices exposed over the network with USB/IP . . . 232
9.34.1 Setting up USB/IP support on a Linux system . . . 232
9.34.2 Security considerations . . . 233

10 Technical background **234**
10.1 Where VirtualBox stores its files . . . 234
10.1.1 Machines created by VirtualBox version 4.0 or later . . . 234
10.1.2 Machines created by VirtualBox versions before 4.0 . . . 235
10.1.3 Global configuration data . . . 235
10.1.4 Summary of 4.0 configuration changes . . . 236
10.1.5 VirtualBox XML files . . . 236
10.2 VirtualBox executables and components . . . 237
10.3 Hardware vs. software virtualization . . . 239
10.4 Paravirtualization providers . . . 240
10.5 Details about software virtualization . . . 240
10.6 Details about hardware virtualization . . . 242
10.7 Nested paging and VPIDs . . . 243

11 VirtualBox programming interfaces **245**

12 Troubleshooting **246**
12.1 Procedures and tools . . . 246
12.1.1 Categorizing and isolating problems . . . 246
12.1.2 Collecting debugging information . . . 247
12.1.3 The built-in VM debugger . . . 247
12.1.4 VM core format . . . 249
12.2 General . . . 250

12.2.1 Guest shows IDE/SATA errors for file-based images on slow host file system . . . 250
12.2.2 Responding to guest IDE/SATA flush requests . . . 251
12.2.3 Performance variation with frequency boosting . . . 251
12.2.4 Frequency scaling effect on CPU usage . . . 251
12.2.5 Inaccurate Windows CPU usage reporting . . . 252
12.2.6 Poor performance caused by host power management . . . 252
12.2.7 GUI: 2D Video Acceleration option is grayed out . . . 252
12.3 Windows guests . . . 252
12.3.1 No USB 3.0 support in Windows 7 guests . . . 252
12.3.2 Windows bluescreens after changing VM configuration . . . 252
12.3.3 Windows 0x101 bluescreens with SMP enabled (IPI timeout) . . . 253
12.3.4 Windows 2000 installation failures . . . 253
12.3.5 How to record bluescreen information from Windows guests . . . 253
12.3.6 PCnet driver failure in 32-bit Windows Server 2003 guests . . . 254
12.3.7 No networking in Windows Vista guests . . . 254
12.3.8 Windows guests may cause a high CPU load . . . 254
12.3.9 Long delays when accessing shared folders . . . 254
12.3.10 USB tablet coordinates wrong in Windows 98 guests . . . 254
12.3.11 Windows guests are removed from an Active Directory domain after restoring a snapshot . . . 255
12.3.12 Restoring d3d8.dll and d3d9.dll . . . 255
12.3.13 Windows 3.x limited to 64 MB RAM . . . 256
12.4 Linux and X11 guests . . . 256
12.4.1 Linux guests may cause a high CPU load . . . 256
12.4.2 AMD Barcelona CPUs . . . 256
12.4.3 Buggy Linux 2.6 kernel versions . . . 257
12.4.4 Shared clipboard, auto-resizing and seamless desktop in X11 guests . . . 257
12.5 Solaris guests . . . 257
12.5.1 Older Solaris 10 releases crash in 64-bit mode . . . 257
12.5.2 Certain Solaris 10 releases may take long to boot with SMP . . . 258
12.5.3 Solaris 8 5/01 and earlier may crash on startup . . . 258
12.6 FreeBSD guests . . . 258
12.6.1 FreeBSD 10.0 may hang with xHCI . . . 258
12.7 Windows hosts . . . 258
12.7.1 VBoxSVC out-of-process COM server issues . . . 258
12.7.2 CD/DVD changes not recognized . . . 259
12.7.3 Sluggish response when using Microsoft RDP client . . . 259
12.7.4 Running an iSCSI initiator and target on a single system . . . 259
12.7.5 Bridged networking adapters missing . . . 260
12.7.6 Host-only networking adapters cannot be created . . . 260
12.8 Linux hosts . . . 260
12.8.1 Linux kernel module refuses to load . . . 260
12.8.2 Linux host CD/DVD drive not found . . . 260
12.8.3 Linux host CD/DVD drive not found (older distributions) . . . 260
12.8.4 Linux host floppy not found . . . 261
12.8.5 Strange guest IDE error messages when writing to CD/DVD . . . 261
12.8.6 VBoxSVC IPC issues . . . 261
12.8.7 USB not working . . . 262
12.8.8 PAX/grsec kernels . . . 262
12.8.9 Linux kernel vmalloc pool exhausted . . . 262
12.9 Solaris hosts . . . 262
12.9.1 Cannot start VM, not enough contiguous memory . . . 262

12.9.2 VM aborts with out of memory errors on Solaris 10 hosts 262

13 Security guide **264**
13.1 General Security Principles 264
13.2 Secure Installation and Configuration 264
13.2.1 Installation Overview 264
13.2.2 Post Installation Configuration 265
13.3 Security Features 265
13.3.1 The Security Model 265
13.3.2 Secure Configuration of Virtual Machines 265
13.3.3 Configuring and Using Authentication 266
13.3.4 Potentially insecure operations 267
13.3.5 Encryption 267

14 Known limitations **269**
14.1 Experimental Features 269
14.2 Known Issues 269

15 Change log **272**
15.1 Version 5.1.22 (2017-04-28) 272
15.2 Version 5.1.20 (2017-04-18) 272
15.3 Version 5.1.18 (2017-03-15) 273
15.4 Version 5.1.16 (2017-03-08) 273
15.5 Version 5.1.14 (2017-01-16) 274
15.6 Version 5.1.12 (2016-12-20) 275
15.7 Version 5.1.10 (2016-11-21) 275
15.8 Version 5.1.8 (2016-10-18) 276
15.9 Version 5.1.6 (2016-09-12) 277
15.10 Version 5.1.4 (2016-08-16) 279
15.11 Version 5.1.2 (2016-07-21) 280
15.12 Version 5.1.0 (2016-07-12) 281
15.13 Version 5.0.24 (2016-06-28) 282
15.14 Version 5.0.22 (2016-06-16) 282
15.15 Version 5.0.20 (2016-04-28) 283
15.16 Version 5.0.18 (2016-04-18) 284
15.17 Version 5.0.16 (2016-03-04) 285
15.18 Version 5.0.14 (2016-01-19) 286
15.19 Version 5.0.12 (2015-12-18) 287
15.20 Version 5.0.10 (2015-11-10) 288
15.21 Version 5.0.8 (2015-10-20) 289
15.22 Version 5.0.6 (2015-10-02) 290
15.23 Version 5.0.4 (2015-09-08) 291
15.24 Version 5.0.2 (2015-08-13) 292
15.25 Version 5.0.0 (2015-07-09) 294
15.26 Version 4.3.28 (2015-05-13) 297
15.27 Version 4.3.26 (2015-03-16) 298
15.28 Version 4.3.24 (2015-03-02) 299
15.29 Version 4.3.22 (2015-02-12) 299
15.30 Version 4.3.20 (2014-11-21) 301
15.31 Version 4.3.18 (2014-10-10) 302
15.32 Version 4.3.16 (2014-09-09) 303
15.33 Version 4.3.14 (2014-07-15) 305
15.34 Version 4.3.12 (2014-05-16) 306
15.35 Version 4.3.10 (2014-03-26) 307

15.36 Version 4.3.8 (2014-02-25) . . . 308
15.37 Version 4.3.6 (2013-12-18) . . . 310
15.38 Version 4.3.4 (2013-11-29) . . . 311
15.39 Version 4.3.2 (2013-11-01) . . . 313
15.40 Version 4.3.0 (2013-10-15) . . . 314
15.41 Older Change log details . . . 317

16 Third-party materials and licenses **318**
16.1 Materials . . . 318
16.2 Licenses . . . 321
16.2.1 GNU General Public License (GPL) . . . 321
16.2.2 GNU Lesser General Public License (LGPL) . . . 325
16.2.3 Mozilla Public License (MPL) . . . 330
16.2.4 MIT License . . . 336
16.2.5 X Consortium License (X11) . . . 336
16.2.6 zlib license . . . 336
16.2.7 OpenSSL license . . . 336
16.2.8 Slirp license . . . 337
16.2.9 liblzf license . . . 338
16.2.10 libpng license . . . 338
16.2.11 lwIP license . . . 338
16.2.12 libxml license . . . 339
16.2.13 libxslt licenses . . . 339
16.2.14 gSOAP Public License Version 1.3a . . . 340
16.2.15 Chromium licenses . . . 345
16.2.16 curl license . . . 347
16.2.17 libgd license . . . 347
16.2.18 BSD license from Intel . . . 348
16.2.19 libjpeg License . . . 348
16.2.20 x86 SIMD extension for IJG JPEG library license . . . 349
16.2.21 FreeBSD license . . . 349
16.2.22 NetBSD license . . . 350
16.2.23 PCRE license . . . 350
16.2.24 libffi license . . . 351
16.2.25 FLTK license . . . 352
16.2.26 Expat license . . . 352
16.2.27 Fontconfig license . . . 352
16.2.28 Freetype license . . . 353
16.2.29 VPX License . . . 355

17 VirtualBox privacy information **356**

Glossary **357**

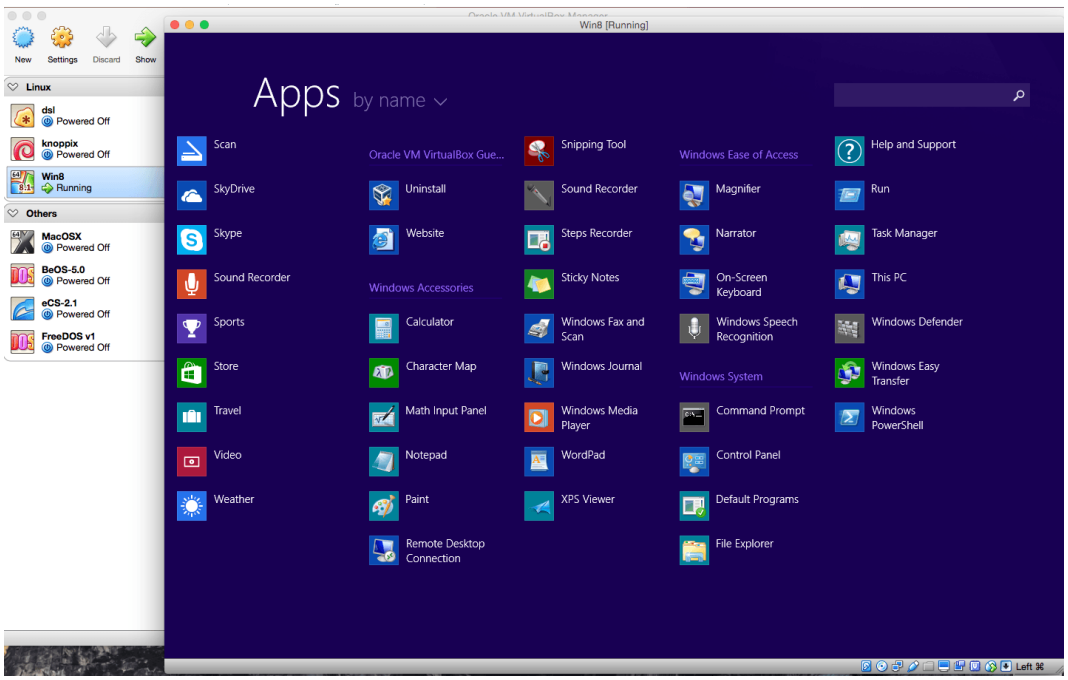
1 First steps

Welcome to Oracle VM VirtualBox!

VirtualBox is a cross-platform virtualization application. What does that mean? For one thing, it installs on your existing Intel or AMD-based computers, whether they are running Windows, Mac, Linux or Solaris operating systems. Secondly, it extends the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. So, for example, you can run Windows and Linux on your Mac, run Windows Server 2008 on your Linux server, run Linux on your Windows PC, and so on, all alongside your existing applications. You can install and run as many virtual machines as you like – the only practical limits are disk space and memory.

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

The following screenshot shows you how VirtualBox, installed on a Mac computer, is running Windows 8 in a virtual machine window:



In this User Manual, we'll begin simply with a quick introduction to virtualization and how to get your first virtual machine running with the easy-to-use VirtualBox graphical user interface. Subsequent chapters will go into much more detail covering more powerful tools and features, but fortunately, it is not necessary to read the entire User Manual before you can use VirtualBox.

You can find a summary of VirtualBox's capabilities in chapter 1.3, *Features overview*, page 13. For existing VirtualBox users who just want to see what's new in this release, there is a detailed list in chapter 15, *Change log*, page 272.

1.1 Why is virtualization useful?

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

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

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

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

  On top of that, with the use of another 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.

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

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

1.2 Some terminology

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

Host operating system (host OS). This is the operating system of the physical computer on which VirtualBox was installed. There are versions of VirtualBox for Windows, Mac OS X, Linux and Solaris hosts; for details, please see chapter 1.4, *Supported host operating systems*, page 15.

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

Guest operating system (guest OS). This is 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 3.1, *Supported guest operating systems*, page 45 for details.

Virtual machine (VM). This is the special environment that VirtualBox creates for your guest operating system while it is running. 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 on another computer.

In a more abstract way, internally, VirtualBox thinks of a VM as a set of parameters that determine its behavior. They include hardware settings (how much memory the VM should have, what hard disks VirtualBox should virtualize through which container files, what CDs are mounted etc.) as well as state information (whether the VM is currently running, saved, its snapshots etc.). These settings are mirrored in the VirtualBox Manager window as well as the `VBoxManage` command line program; see chapter 8, *VBoxManage*, page 114. In other words, a VM is also what you can see in its settings dialog.

Guest Additions. This refers to special software packages which are shipped with VirtualBox but 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, *Guest Additions*, page 60.

1.3 Features overview

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

- **Portability.** VirtualBox runs on a large number of 32-bit and 64-bit host operating systems (again, see chapter 1.4, *Supported host operating systems*, page 15 for details).

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

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

  In addition, virtual machines can easily be imported and exported using the Open Virtualization Format (OVF, see chapter 1.14, *Importing and exporting virtual machines*, page 31), an industry standard created for this purpose. You can even import OVFs that were created with a different virtualization software.

- **No hardware virtualization required.** For many scenarios, VirtualBox does not require the processor features built into newer hardware like Intel VT-x or AMD-V. As opposed to many other virtualization solutions, you can therefore use VirtualBox even on older hardware where these features are not present. The technical details are explained in chapter 10.3, *Hardware vs. software virtualization*, page 239.

- **Guest Additions: shared folders, seamless windows, 3D virtualization.** 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, accelerated 3D graphics and more. The Guest Additions are described in detail in chapter 4, *Guest Additions*, page 60.

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 4.3, *Shared folders*, page 68.

- **Great hardware support.** Among others, VirtualBox supports:
  - **Guest multiprocessing (SMP).** VirtualBox can present up to 32 virtual CPUs to each virtual machine, irrespective of how many CPU cores are physically present on your host.
  - **USB device support.** 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 3.10.1, *USB settings*, page 56.
  - **Hardware compatibility.** VirtualBox virtualizes a vast array of virtual devices, among them many devices that are typically provided by other virtualization platforms. That includes IDE, SCSI and SATA hard disk controllers, several virtual network cards and sound cards, virtual serial and parallel ports and an Input/Output Advanced Programmable Interrupt Controller (I/O APIC), which is found in many modern PC systems. This eases cloning of PC images from real machines and importing of third-party virtual machines into VirtualBox.
  - **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 full screen modes).
  - **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 5.10, *iSCSI servers*, page 91.
  - **PXE Network boot.** The integrated virtual network cards of VirtualBox fully support remote booting via the Preboot Execution Environment (PXE).
- **Multigeneration branched snapshots.** VirtualBox can save arbitrary snapshots of the state of the virtual machine. You can go back in time and revert the virtual machine to any such snapshot and start an alternative VM configuration from there, effectively creating a whole snapshot tree. For details, see chapter 1.10, *Snapshots*, page 26. You can create and delete snapshots while the virtual machine is running.
- **VM groups.** VirtualBox provides a groups feature that enables the user to organize and control virtual machines collectively, as well as individually. In addition to basic groups, it is also possible for any VM to be in more than one group, and for groups to be nested in a hierarchy – i.e. groups of groups. In general, the operations that can be performed on groups are the same as those that can be applied to individual VMs i.e. Start, Pause, Reset, Close (Save state, Send Shutdown, Poweroff), Discard Saved State, Show in fileSystem, Sort.
- **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 1.16, *Alternative front-ends*, page 32 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 11, *VirtualBox programming interfaces*, page 245 for details.

- **Remote machine display.** The VirtualBox Remote Desktop Extension (VRDE) allows for high-performance remote access to any running virtual machine. This extension supports the Remote Desktop Protocol (RDP) originally built into Microsoft Windows, with special additions for full client USB support.

  The VRDE does not rely on the RDP server that is built into Microsoft Windows; instead, it is plugged directly into the virtualization layer. As a result, it works with guest operating systems other than Windows (even in text mode) and does not require application support in the virtual machine either. The VRDE is described in detail in chapter 7.1, *Remote display (VRDP support)*, page 104.

  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 7.1.5, *RDP authentication*, page 108 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 7.1.4, *Remote USB*, page 108 for details.

1.4 Supported host operating systems

Currently, VirtualBox runs on the following host operating systems:

- **Windows** hosts:[1]
  - Windows Vista SP1 and later (32-bit and 64-bit)
  - Windows Server 2008 (64-bit)
  - Windows Server 2008 R2 (64-bit)
  - Windows 7 (32-bit and 64-bit)
  - Windows 8 (32-bit and 64-bit)
  - Windows 8.1 (32-bit and 64-bit)
  - Windows 10 RTM build 10240 (32-bit and 64-bit)
  - Windows Server 2012 (64-bit)
  - Windows Server 2012 R2 (64-bit)
- **Mac OS X** hosts (64-bit):[2]
  - 10.9 (Mavericks)

[1] Support for 64-bit Windows was added with VirtualBox 1.5. Support for Windows XP was removed with VirtualBox 5.0.

[2] Preliminary Mac OS X support (beta stage) was added with VirtualBox 1.4, full support with 1.6. Mac OS X 10.4 (Tiger) support was removed with VirtualBox 3.1. Support for Mac OS X 10.7 (Lion) and earlier was removed with VirtualBox 5.0. Support for Mac OS X 10.8 (Mountain Lion) was removed with VirtualBox 5.1.

  - 10.10 (Yosemite)
  - 10.11 (El Capitan)
  - 10.12 (Sierra)

  Intel hardware is required; please see chapter 14, *Known limitations*, page 269 also.

- **Linux** hosts (32-bit and 64-bit[3]). Among others, this includes:
  - Ubuntu 12.04 to 16.10
  - Debian GNU/Linux 7 ("Wheezy") and 8 ("Jessie")
  - Oracle Enterprise Linux 5, Oracle Linux 6 and 7
  - Redhat Enterprise Linux 5, 6 and 7
  - Fedora Core / Fedora 6 to 25
  - Gentoo Linux
  - openSUSE 11.4 to 13.2

  It should be possible to use VirtualBox on most systems based on Linux kernel 2.6 or 3.x using either the VirtualBox installer or by doing a manual installation; see chapter 2.3, *Installing on Linux hosts*, page 37. However, the formally tested and supported Linux distributions are those for which we offer a dedicated package.

  Note that starting with VirtualBox 2.1, Linux 2.4-based host operating systems are no longer supported.

- **Solaris** hosts (64-bit only) are supported with the restrictions listed in chapter 14, *Known limitations*, page 269:
  - Solaris 11
  - Solaris 10 (U10 and higher)

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

1.5 Installing VirtualBox and extension packs

VirtualBox comes in many different packages, and installation depends on your host operating system. If you have installed software before, installation should be straightforward: on each host platform, VirtualBox uses the installation method that is most common and easy to use. If you run into trouble or have special requirements, please refer to chapter 2, *Installation details*, page 34 for details about the various installation methods.

Starting with version 4.0, VirtualBox is split into several components.

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

2. Additional extension packs can be downloaded which extend the functionality of the VirtualBox base package. Currently, Oracle provides the one extension pack, which can be found at `http://www.virtualbox.org` and provides the following added functionality:

   a) The virtual USB 2.0 (EHCI) device; see chapter 3.10.1, *USB settings*, page 56.

   b) The virtual USB 3.0 (xHCI) device; see chapter 3.10.1, *USB settings*, page 56.

[3] Support for 64-bit Linux was added with VirtualBox 1.4.

c) VirtualBox Remote Desktop Protocol (VRDP) support; see chapter 7.1, *Remote display (VRDP support)*, page 104.

d) Host webcam passthrough; see chapter chapter 9.7.1, *Using a host webcam in the guest*, page 195.

e) Intel PXE boot ROM.

f) Experimental support for PCI passthrough on Linux hosts; see chapter 9.6, *PCI passthrough*, page 194.

g) Disk image encryption with AES algorithm; see chapter 9.31, *Encryption of disk images*, page 226.

VirtualBox extension packages have a `.vbox-extpack` file name extension. To install an extension, simply double-click on the package file and a Network Operations Manager window will appear, guiding you through the required steps.

To view the extension packs that are currently installed, please start the VirtualBox Manager (see the next section). From the "File" menu, please select "Preferences". In the window that shows up, go to the "Extensions" category which shows you the extensions which are currently installed and allows you to remove a package or add a new one.

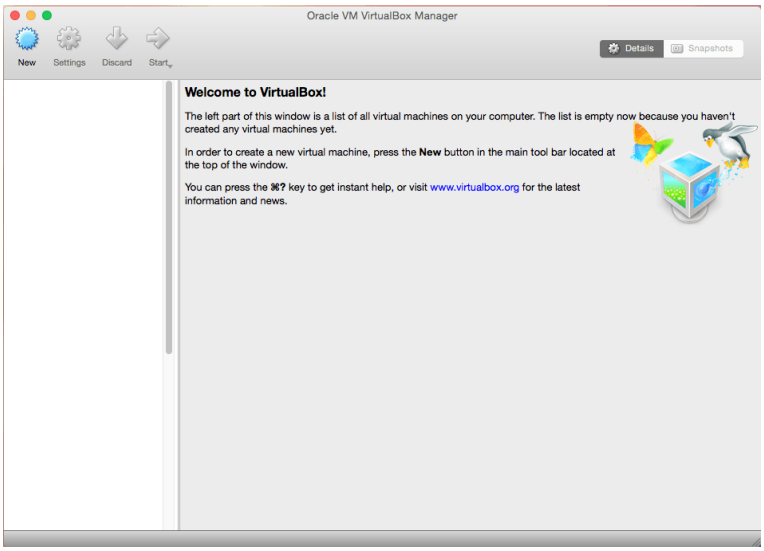
Alternatively you can use VBoxManage on the command line: see chapter 8.41, *VBoxManage extpack*, page 183 for details.

1.6 Starting VirtualBox

After installation, you can start VirtualBox as follows:

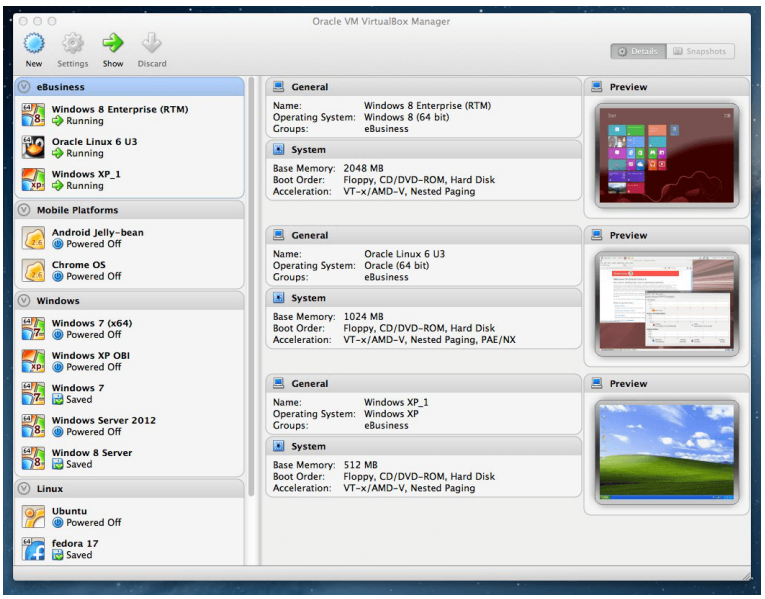
- On a Windows host, in the standard "Programs" menu, click on the item in the "VirtualBox" group. On Vista or Windows 7, you can also type "VirtualBox" in the search box of the "Start" menu.
- On a Mac OS X host, in the Finder, double-click on the "VirtualBox" item in the "Applications" folder. (You may want to drag this item onto your Dock.)
- On a Linux or 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.

When you start VirtualBox for the first time, a window like the following should come up:



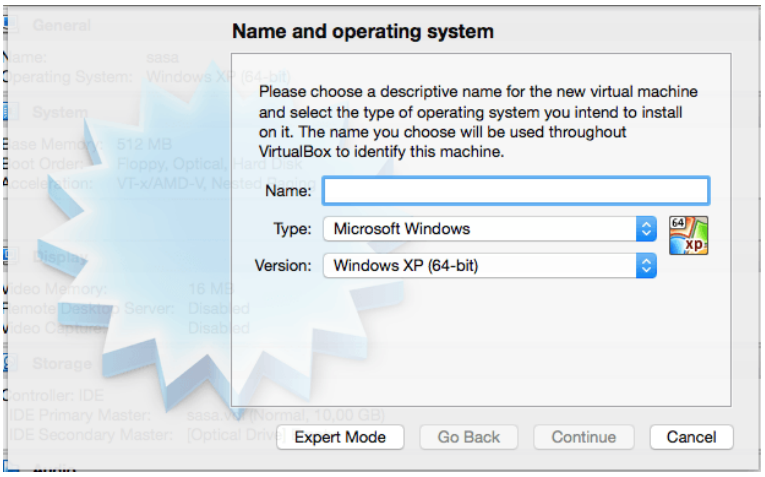
This window is called the **"VirtualBox Manager".** On the left, you can see a pane that will later list all your virtual machines. Since you have not created any, the list is empty. A row of buttons above it allows you to create new VMs and work on existing VMs, once you have some. The pane on the right displays the properties of the virtual machine currently selected, if any. Again, since you don't have any machines yet, the pane displays a welcome message.

To give you an idea what VirtualBox might look like later, after you have created many machines, here's another example:



1.7 Creating your first virtual machine

Click on the "New" button at the top of the VirtualBox Manager window. A wizard will pop up to guide you through setting up a new virtual machine (VM):



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

1. The **VM name** will later be shown in the VM list of the VirtualBox Manager window, and it will be used for the VM's files on disk. Even though any name could be used, keep in

mind that once you have created a few VMs, you will appreciate if you have given your VMs rather informative names; "My VM" would thus be less useful than "Windows XP SP2 with OpenOffice".

2. For **"Operating System Type",** select the operating system that you want to install later. The supported operating systems are grouped; if you want to install something very unusual that is not listed, select "Other". Depending on your selection, VirtualBox will enable or disable certain VM settings that your guest operating system may require. This is particularly important for 64-bit guests (see chapter 3.1.2, *64-bit guests*, page 46). It is therefore recommended to always set it to the correct value.

3. On the next page, select the **memory (RAM)** that VirtualBox should allocate every time the virtual machine is started. The amount of memory given here will be taken away from your host machine and presented to the guest 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.

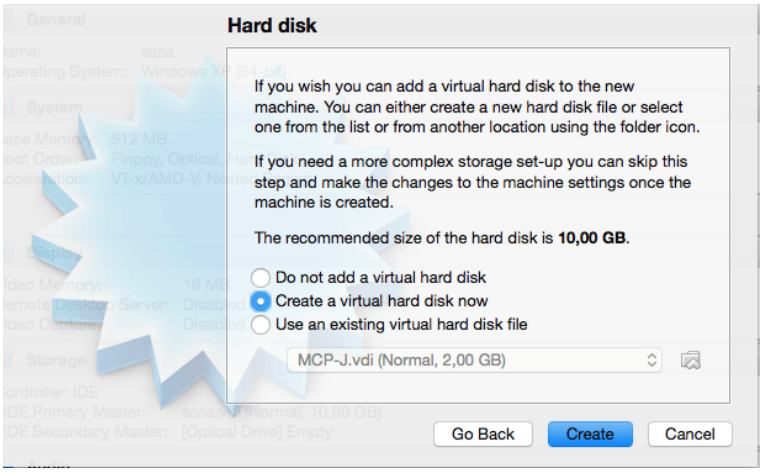
   So, as a rule of thumb, if you have 1 GB of RAM or more in your host computer, it is usually safe to allocate 512 MB to each VM. But, in any case, make sure you always have at least 256 to 512 MB of RAM left on your host operating system. Otherwise you may cause your host OS to excessively swap out memory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

4. Next, you must specify a **virtual hard disk** for your VM.

   There are many and potentially complicated ways in which VirtualBox can provide hard disk space to a VM (see chapter 5, *Virtual storage*, page 79 for details), 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. This file represents an entire hard disk then, so you can even copy it to another host and use it with another VirtualBox installation.

   The wizard shows you the following window:



Here you have the following options:

- To create a new, empty virtual hard disk, press the **"New"** button.
- You can pick an **existing** disk image file.

  The **drop-down list** presented in the window contains all disk images which are currently remembered by VirtualBox, probably because they are currently attached to a virtual machine (or have been in the past).

  Alternatively, you can click on the small **folder button** next to the drop-down list to bring up a standard file dialog, which allows you to pick any disk image file on your host disk.

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.

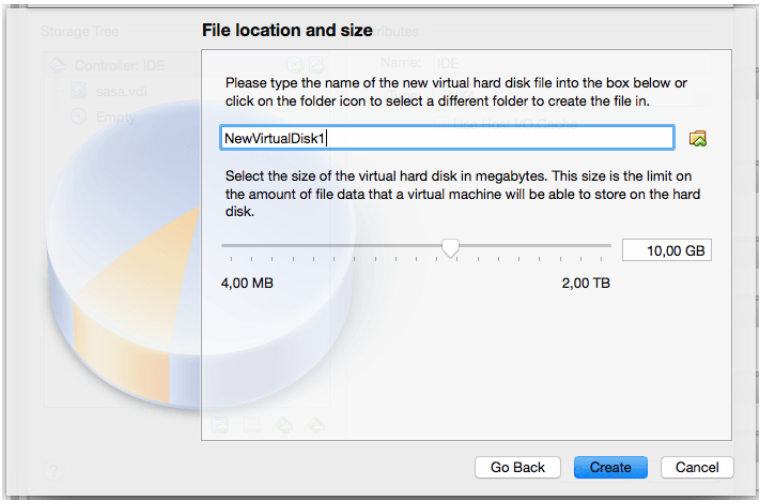
This brings up another window, the **"Create New Virtual Disk Wizard"**, which helps you create a new disk image file in the new virtual machine's folder.

VirtualBox supports two types of image files:

- A **dynamically allocated 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 allocated file.

For details about the differences, please refer to chapter 5.2, *Disk image files (VDI, VMDK, VHD, HDD)*, page 82.

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. The limit of the image file size can be changed later (see chapter 8.23, *VBoxManage modifymedium*, page 156 for details).



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

5. 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 Manager window, with the name you entered initially.

> **Note:** After becoming familiar with the use of wizards, consider using the Expert Mode available in some wizards. Where available, this is selectable using a button, and speeds up user processes using wizards.

1.8 Running your virtual machine

To start a virtual machine, you have several options:

- Double-click on its entry in the list within the Manager window or
- select its entry in the list in the Manager window it and press the "Start" button at the top or
- for virtual machines created with VirtualBox 4.0 or later, navigate to the "VirtualBox VMs" folder in your system user's home directory, find the subdirectory of the machine you want to start and double-click on the machine settings file (with a `.vbox` file extension).

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 1.2, *Some terminology*, page 12.

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

1.8.1 Starting a new VM for the first time

When a VM gets started for the first time, another wizard – the **"First Start Wizard"** – will pop up to help you select an **installation medium**. Since the VM is created empty, it would otherwise behave just like a real computer with no operating system installed: it will do nothing and display an error message that no bootable operating system was found.

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

- If you have physical 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 wizard's drop-down list of installation media, 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.

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

  If your medium is not in the list (especially if you are using VirtualBox for the first time), select the small folder icon next to the drop-down list to bring up a standard file dialog, with which you can pick the image file on your host disks.

In both cases, after making the choices in the wizard, you will be able to install your operating system.

1.8.2 Capturing and releasing keyboard and mouse

As of version 3.2, VirtualBox provides a virtual USB tablet device to new virtual machines through which mouse events are communicated to the guest operating system. As a result, if you are running a modern guest operating system that can handle such devices, mouse support may work out of the box without the mouse being "captured" as described below; see chapter 3.4.1, *"Motherboard" tab*, page 49 for more information.

Otherwise, if the virtual machine only sees standard PS/2 mouse and keyboard devices, 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 guest 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* on your keyboard; on a Mac host, the default host key is the left Command key. You can change this default in the VirtualBox Global Settings, see chapter 1.15, *Global Settings*, page 32. 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 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 re-enable the host keyboard, all key presses will go through the host again, so that sequences like Alt-Tab will no longer reach the guest. For technical reasons it may not be possible for the VM to get all keyboard input even when it does own the keyboard. Examples of this are the Ctrl-Alt-Del sequence on Windows hosts or single keys grabbed by other applications on X11 hosts like the GNOME desktop's "Control key highlights mouse pointer" functionality.

- 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, *Guest Additions*, page 60.

1.8.3 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, on Linux and Solaris hosts, which use the X Window System, 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, on Linux hosts supporting virtual terminals, the key combination **Ctrl+Alt+Fx** (where Fx is one of the function keys from F1 to F12) normally 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 "Input" → "Keyboard" 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 or Solaris 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 or Solaris 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" → "Preferences" → "Input" → "Auto-capture keyboard".

1.8.4 Changing removable media

While a virtual machine is running, you can change removable media in the "Devices" menu of the VM's window. Here you can select in detail what 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 the "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 1.11, *Virtual machine configuration*, page 29.

1.8.5 Resizing the machine's window

You can resize the virtual machine's window when it is running. In that case, one of three things will happen:

1. If you have **"scale mode"** enabled, then the virtual machine's screen will be scaled to the size of the window. This can be useful if you have many machines running and want to have a look at one of them while it is running in the background. Alternatively, it might be useful to enlarge a window if the VM's output screen is very small, for example because you are running an old operating system in it.

   To enable scale mode, press the **host key + C**, or select "Scale mode" from the "Machine" menu in the VM window. To leave scale mode, press the host key + C again.

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

   Please see chapter 14, *Known limitations*, page 269 for additional remarks.

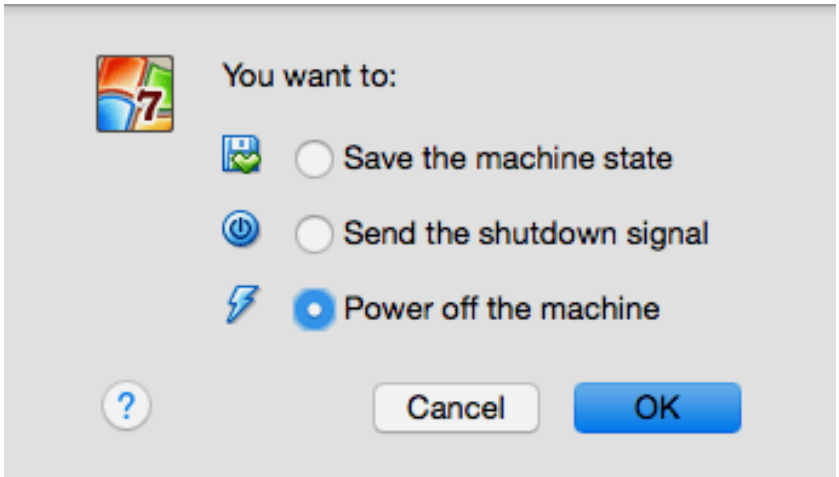
2. If you have the Guest Additions installed and they support automatic **resizing**, the Guest Additions will automatically adjust the screen resolution of the guest operating system. For example, if you are running a Windows guest with a resolution of 1024x768 pixels and you then resize the VM window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

   Please see chapter 4, *Guest Additions*, page 60 for more information about the Guest Additions.

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

1.8.6 Saving the state of the machine

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



The difference between these three 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 start the VM again later, you will find that the VM continues exactly where it was left off. All your programs will still be open, and your computer resumes operation. Saving the state of a virtual machine is thus in some ways similar to suspending a laptop computer (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 the VM is running a fairly modern operating system, this should trigger a proper shutdown mechanism from within the VM.

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

  > **Warning:** This is equivalent to pulling the power plug on a real computer without shutting it down properly. If you start the machine again after powering it off, your 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.

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

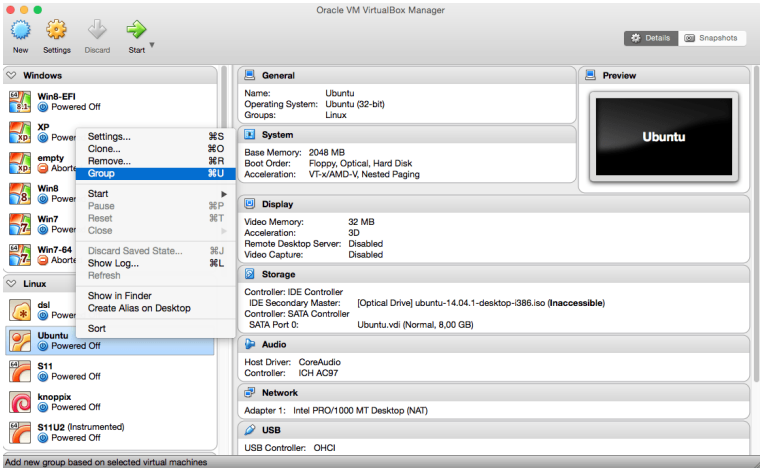
The **"Discard"** button in the VirtualBox Manager window discards a virtual machine's saved state. This has the same effect as powering it off, and the same warnings apply.

1.9 Using VM groups

VM groups enable the user to create ad hoc groups of VMs, and to manage and perform functions on them collectively, as well as individually. There are a number of features relating to groups:

1. Create a group using GUI option 1) Drag one VM on top of another VM.

   Create a group using GUI option 2) Select multiple VMs and select "Group" on the right click menu, as follows:



2. Command line option 1) Create a group and assign a VM:

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

   creates a group "TestGroup" and attaches the VM "Fred" to that group.

   Command line option 2) Detach a VM from the group, and delete the group if empty:

   ```
   VBoxManage modifyvm "Fred" --groups ""
   ```

   It detaches all groups from the VM "Fred" and deletes the empty group.

3. Multiple groups e.g.:

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

   It creates the groups "TestGroup" and "TestGroup2" (if they don't exist yet) and attaches the VM "Fred" to both of them.

4. Nested groups – hierarchy of groups e.g.:

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

   It attaches the VM "Fred" to the subgroup "TestGroup2" of the "TestGroup" group.

5. Summary of group commands: Start, Pause, Reset, Close (save state, send shutdown signal, poweroff), Discard Saved State, Show in File System, Sort.

1.10 Snapshots

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

You can see the snapshots of a virtual machine by first selecting a machine in the VirtualBox Manager and then clicking on the “Snapshots” button at the top right. Until you take a snapshot of the machine, the list of snapshots will be empty except for the “Current state” item, which represents the “Now” point in the lifetime of the virtual machine.

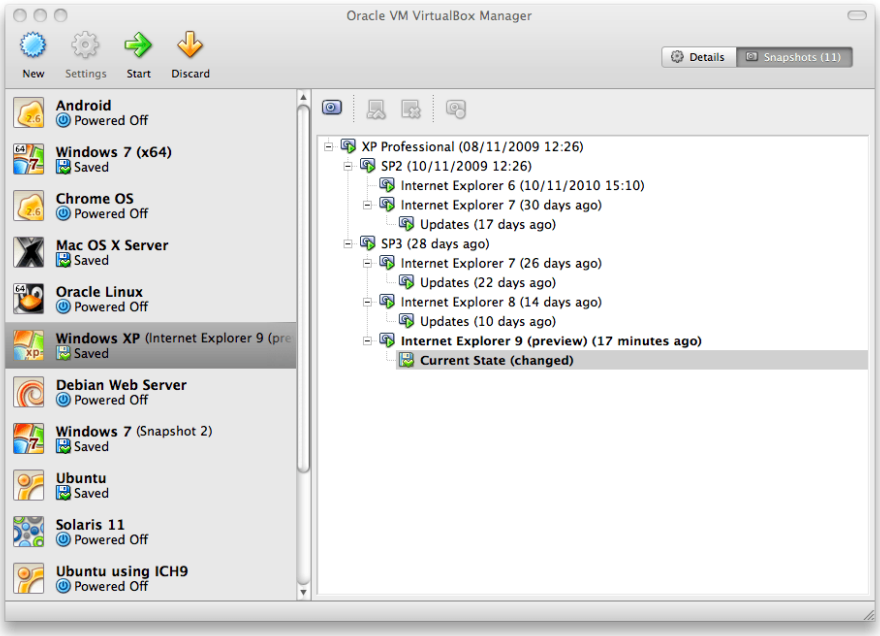
1.10.1 Taking, restoring and deleting snapshots

There are three operations related to snapshots:

1. You can **take a snapshot**. This makes a copy of the machine’s current state, to which you can go back at any given time later.
   - If your VM is currently running, select “Take snapshot” from the “Machine” pull-down menu of the VM window.
   - 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
     - either on the small camera icon (for “Take snapshot”) or
     - right-click on the “Current State” item in the list and select “Take snapshot” from the menu.

   In any case, a window will pop up and ask you for a snapshot name. This name is purely for reference purposes to help you remember the state of the snapshot. For example, a useful name would be “Fresh installation from scratch, no Guest Additions”, or “Service Pack 3 just installed”. You can also add a longer text in the “Description” field if you want.

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



VirtualBox imposes no limits on the number of snapshots you can take. The only practical limitation is disk space on your host: each snapshot stores the state of the virtual machine and thus occupies some disk space. (See the next section for details on what exactly is stored in a snapshot.)

2. You can **restore a snapshot** by right-clicking on any snapshot you have taken in the list of snapshots. By restoring a snapshot, you go back (or forward) in time: the current state of the machine is lost, and the machine is restored to the exact state it was in when the snapshot was taken.[4]

> **Note:** Restoring a snapshot will affect the virtual hard drives that are connected to your VM, as the entire state of the virtual hard drive will be reverted as well. This means also that all files that have been created since the snapshot and all other file changes *will be lost*. In order to prevent such data loss while still making use of the snapshot feature, it is possible to add a second hard drive in "write-through" mode using the `VBoxManage` interface and use it to store your data. As write-through hard drives are *not* included in snapshots, they remain unaltered when a machine is reverted. See chapter 5.4, *Special image write modes*, page 84 for details.

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

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

3. You can also **delete a snapshot**, which will not affect the state of the virtual machine, but only release the files on disk that VirtualBox used to store the snapshot data, thus freeing disk space. To delete a snapshot, right-click on it in the snapshots tree and select "Delete". As of VirtualBox 3.2, snapshots can be deleted even while a machine is running.

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

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

1.10.2 Snapshot contents

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

- It contains a complete copy of the VM settings, including the hardware configuration, so that when you restore a snapshot, the VM settings are restored as well. (For example, if

[4] Both the terminology and the functionality of restoring snapshots has changed with VirtualBox 3.1. Before that version, it was only possible to go back to the very last snapshot taken – not earlier ones, and the operation was called "Discard current state" instead of "Restore last snapshot". The limitation has been lifted with version 3.1. It is now possible to restore *any* snapshot, going backward and forward in time.

you changed the hard disk configuration or the VM's system settings, that change is undone when you restore the snapshot.)

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

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

  (Strictly speaking, this is only true for virtual hard disks in "normal" mode. As mentioned above, you can configure disks to behave differently with snapshots; see chapter 5.4, *Special image write modes*, page 84. Even more formally and technically correct, it is not the virtual disk itself that is restored when a snapshot is restored. Instead, when a snapshot is taken, VirtualBox creates differencing images which contain only the changes since the snapshot were taken, and when the snapshot is restored, VirtualBox throws away that differencing image, thus going back to the previous state. This is both faster and uses less disk space. For the details, which can be complex, please see chapter 5.5, *Differencing images*, page 86.)

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

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

  The memory state file can be as large as the memory size of the virtual machine and will therefore occupy quite some disk space as well.

1.11 Virtual machine configuration

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

Clicking on the "Settings" button in the toolbar at the top brings up a detailed window where you can configure many of the properties of the selected VM. 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, *Configuring virtual machines*, page 45. Even more parameters are available with the VirtualBox command line interface; see chapter 8, *VBoxManage*, page 114.

1.12 Removing virtual machines

To remove a virtual machine which you no longer need, right-click on it in the Manager's VM list select "Remove" from the context menu that comes up.

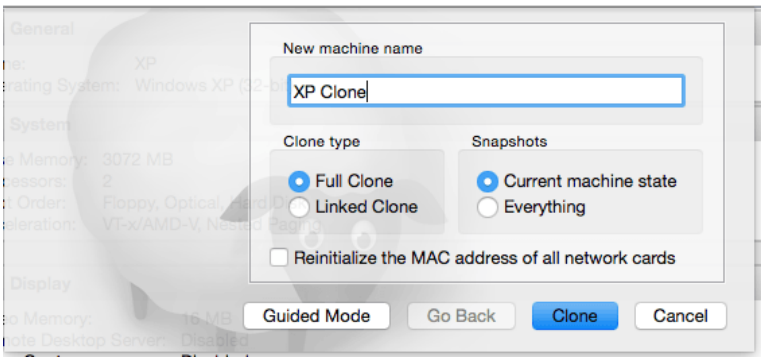
A confirmation window will come up that allows you to select whether the machine should only be removed from the list of machines or whether the files associated with it should also be deleted.

The "Remove" menu item is disabled while a machine is running.

1.13 Cloning virtual machines

To experiment with a VM configuration, test different guest OS levels or to simply backup a VM, VirtualBox can create a full or a linked copy of an existing VM.[5]

A wizard will guide you through the clone process:



This wizard can be invoked from the context menu of the Manager's VM list (select "Clone") or the "Snapshots" view of the selected VM. First choose a new name for the clone. When you select **Reinitialize the MAC address of all network cards** every network card get a new MAC address assigned. This is useful when both, the source VM and the cloned VM, have to operate on the same network. If you leave this unchanged, all network cards have the same MAC address like the one in the source VM. Depending on how you invoke the wizard you have different choices for the cloning operation. First you need to decide if the clone should be linked to the source VM or a fully independent clone should be created:

- **Full clone:** In this mode all depending disk images are copied to the new VM folder. The clone can fully operate without the source VM.
- **Linked clone:** In this mode new differencing disk images are created where the parent disk images are the source disk images. If you selected the current state of the source VM as clone point, a new snapshot will be created implicitly.

After selecting the clone mode, you need to decide about what exactly should be cloned. You can always create a clone of the *current state* only or *all*. When you select *all*, the current state and in addition all snapshots are cloned. Have you started from a snapshot which has additional children, you can also clone the *current state and all children*. This creates a clone starting with this snapshot and includes all child snapshots.

The clone operation itself can be a lengthy operation depending on the size and count of the attached disk images. Also keep in mind that every snapshot has differencing disk images attached, which need to be cloned as well.

The "Clone" menu item is disabled while a machine is running.

For how to clone a VM at the command line, please see chapter 8.9, *VBoxManage clonevm*, page 140.

[5] Cloning support was introduced with VirtualBox 4.1.

1.14 Importing and exporting virtual machines

VirtualBox can import and export virtual machines in the industry-standard Open Virtualization Format (OVF).[6]

OVF is a cross-platform standard supported by many virtualization products which allows for creating ready-made virtual machines that can then be imported into a virtualizer such as VirtualBox. VirtualBox makes OVF import and export easy to access and supports it from the Manager window as well as its command-line interface. This allows for packaging so-called **virtual appliances**: disk images together with configuration settings that can be distributed easily. This way one can offer complete ready-to-use software packages (operating systems with applications) that need no configuration or installation except for importing into VirtualBox.

> **Note:** The OVF standard is complex, and support in VirtualBox is an ongoing process. In particular, no guarantee is made that VirtualBox supports all appliances created by other virtualization software. For a list of known limitations, please see chapter 14, *Known limitations*, page 269.

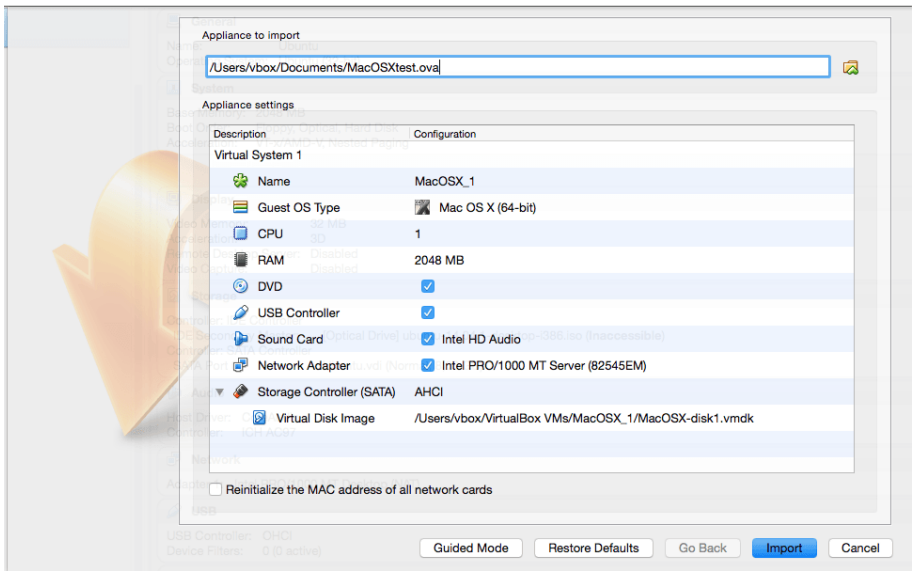
Appliances in OVF format can appear in two variants:

1. They can come in several files, as one or several disk images, typically in the widely-used VMDK format (see chapter 5.2, *Disk image files (VDI, VMDK, VHD, HDD)*, page 82) and a textual description file in an XML dialect with an `.ovf` extension. These files must then reside in the same directory for VirtualBox to be able to import them.

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

To **import** an appliance in one of the above formats, simply double-click on the OVF/OVA file.[7] Alternatively, select "File" → "Import appliance" from the Manager window. In the file dialog that comes up, navigate to the file with either the `.ovf` or the `.ova` file extension.

If VirtualBox can handle the file, a dialog similar to the following will appear:



[6] OVF support was originally introduced with VirtualBox 2.2 and has seen major improvements with every version since.
[7] Starting with version 4.0, VirtualBox creates file type associations for OVF and OVA files on your host operating system.

This presents the virtual machines described in the OVF file and allows you to change the virtual machine settings by double-clicking on the description items. Once you click on **"Import"**, VirtualBox will copy the disk images and create local virtual machines with the settings described in the dialog. These will then show up in the Manager's list of virtual machines.

Note that since disk images tend to be big, and VMDK images that come with virtual appliances are typically shipped in a special compressed format that is unsuitable for being used by virtual machines directly, the images will need to be unpacked and copied first, which can take a few minutes.

For how to import an image at the command line, please see chapter 8.10, *VBoxManage import*, page 141.

Conversely, to **export** virtual machines that you already have in VirtualBox, select "File" → "Export appliance". A different dialog window shows up that allows you to combine several virtual machines into an OVF appliance. Then, select the target location where the target files should be stored, and the conversion process begins. This can again take a while.

For how to export an image at the command line, please see chapter 8.11, *VBoxManage export*, page 142.

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

1.15 Global Settings

The global settings dialog can be reached through the **File** menu, selecting the **Preferences...** item. It offers a selection of settings which apply to all virtual machines of the current user or in the case of **Extensions** to the entire system:

1. **General** Enables the user to specify the default folder/directory for VM files, and the VRDP Authentication Library.
2. **Input** Enables the user to specify the Host Key. It identifies the key that toggles whether the cursor is in the focus of the VM or the Host operating system windows (see chapter 1.8.2, *Capturing and releasing keyboard and mouse*, page 22) and which is also used to trigger certain VM actions (see chapter 1.8.3, *Typing special characters*, page 23)
3. **Update** Enables the user to specify various settings for Automatic Updates.
4. **Language** Enables the user to specify the GUI language.
5. **Display** Enables the user to specify the screen resolution, and its width and height.
6. **Network** Enables the user to configure the details of Host Only Networks.
7. **Extensions** Enables the user to list and manage the installed extension packages.
8. **Proxy** Enables the user to configure a HTTP Proxy Server.

1.16 Alternative front-ends

As briefly mentioned in chapter 1.3, *Features overview*, page 13, VirtualBox has a very flexible internal design that allows for using multiple interfaces to control the same virtual machines. To illustrate, you can, for example, start a virtual machine with the VirtualBox Manager window

and then stop it from the command line. With VirtualBox's support for the Remote Desktop Protocol (RDP), you can even run virtual machines remotely on a headless server and have all the graphical output redirected over the network.

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

1. `VirtualBox` is the VirtualBox Manager. This graphical user interface uses the Qt toolkit; most of this User Manual is dedicated to describing it. While this is the easiest to use, some of the more advanced VirtualBox features are kept away from it to keep it simple.

2. `VBoxManage` is our command-line interface for automated and very detailed control of every aspect of VirtualBox. It is described in chapter 8, *VBoxManage*, page 114.

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 9.1, *VBoxSDL, the simplified VM displayer*, page 185.

4. Finally, `VBoxHeadless` is yet another front-end that produces no visible output on the host at all, but can act as a RDP server if the VirtualBox Remote Desktop Extension (VRDE) is installed and enabled for the VM. As opposed to the other graphical interfaces, the headless front-end requires no graphics support. This is useful, for example, if you want to host your virtual machines on a headless Linux server that has no X Window system installed. For details, see chapter 7.1.2, *VBoxHeadless, the remote desktop server*, page 105.

If the above front-ends still do not satisfy your particular needs, it is possible to create yet another front-end to the complex virtualization engine that is the core of VirtualBox, as the VirtualBox core neatly exposes all of its features in a clean API; please refer to chapter 11, *VirtualBox programming interfaces*, page 245.

2 Installation details

As installation of VirtualBox varies depending on your host operating system, we provide installation instructions in four separate chapters for Windows, Mac OS X, Linux and Solaris, 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 1.4, *Supported host operating systems*, page 15.

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 executable file (contains both 32- and 64-bit architectures)
- or by entering

  ```
  VirtualBox.exe -extract
  ```

  on the command line. This will extract both installers into a temporary directory in which you'll then find the usual .MSI files. Then you can do a

  ```
  msiexec /i VirtualBox-<version>-MultiArch_<x86|amd64>.msi
  ```

  to perform the installation.

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 Bridged Networking (to make your VM's virtual network cards accessible from other machines on your physical network).

Python Support This package contains Python scripting support for the VirtualBox API (see chapter 11, *VirtualBox programming interfaces*, page 245). For this to work, an already working Windows Python installation on the system is required.

> **Note:** Python version $\geq$ 2.6 is required. Since VirtualBox 5.1 Python 3 is also supported.

[1]

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 Windows "Start" menu 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 by first extracting it by using

```
VirtualBox.exe -extract
```

and then do as follows:

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

or

```
msiexec /i VirtualBox-<version>-MultiArch_<x86|amd64>.msi ALLUSERS=2
```

on the extracted .MSI files. This will install VirtualBox only for the current user.

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

VBoxApplication Main binaries of VirtualBox.

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

VBoxUSB USB support.

VBoxNetwork All networking support; includes the VBoxNetworkFlt and VBoxNetworkAdp features (see below).

VBoxNetworkFlt Bridged networking support.

VBoxNetworkAdp Host-only networking support.

VBoxPython Python support.

> **Note:** Python version $\geq$ 2.6 is required. Since VirtualBox 5.1 Python 3 is also supported.

For example, to only install USB support along with the main binaries, do a:

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

or

```
msiexec /i VirtualBox-<version>-MultiArch_<x86|amd64>.msi ADDLOCAL=VBoxApplication,VBoxUSB
```

The user is able to choose between NDIS5 and NDIS6 host network filters drivers during the installation. This is realized via a command line parameter `NETWORKTYPE`. The NDIS6 driver is default for Windows Vista and later. For older Windows versions, the installer will automatically select the NDIS5 driver and this cannot be changed. For Windows Vista and later the user can force to install the (legacy) NDIS5 host network filter driver using `NETWORKTYPE=NDIS5`. For example, to install the NDIS5 driver on Windows 7, do

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

or

```
msiexec /i VirtualBox-<version>-MultiArch_<x86|amd64>.msi NETWORKTYPE=NDIS5
```

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

2.1.3 Uninstallation

As VirtualBox uses the standard Microsoft Windows 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.

2.1.5 Public properties

The following public properties can be specified via MSI API,

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

or

```
msiexec /i VirtualBox-<version>-MultiArch_<x86|amd64>.msi NAME=VALUE [...]
```

to control additional behavior and/or features of the Windows host installer:

VBOX_INSTALLDESKTOPSHORTCUT Specifies whether or not a VirtualBox icon on the desktop should be created.

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

VBOX_INSTALLQUICKLAUNCHSHORTCUT Specifies whether or not a VirtualBox icon in the Quick Launch Bar should be created.

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

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

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

VBOX_START Specifies whether or not VirtualBox should be started right after successful installation.

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

2.2 Installing on Mac OS X hosts

2.2.1 Performing the installation

For Mac OS X hosts, 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 or use the following command line:

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

Then open a terminal session and execute:

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

2.3 Installing on Linux hosts

2.3.1 Prerequisites

For the various versions of Linux that we support as host operating systems, please refer to chapter 1.4, *Supported host operating systems*, page 15.

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 5.3.2 or higher (Qt 5.6.2 or higher recommended);
- 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`, the graphical VirtualBox manager, requires both Qt and SDL; `VBoxSDL`, our simplified GUI, requires only SDL. By contrast, if you only want to run `VBoxHeadless`, neither Qt nor SDL are required.

2.3.2 The VirtualBox driver modules

In order to run other operating systems in virtual machines alongside your main operating system, VirtualBox needs to integrate very tightly into the system. To do this it installs a "driver" module called `vboxdrv` which does a lot of that work into the system kernel, which is the part of the operating system which controls your processor and physical hardware. Without this kernel module, you can still use the VirtualBox manager to configure virtual machines, but they will not start. It also installs network drivers called `vboxnetflt` and `vboxnetadp` which let virtual machines make more use of your computer's network capabilities and are needed for any virtual machine networking beyond the basic "NAT" mode.

Since distributing driver modules separately from the kernel is not something which Linux supports well, we create the modules on the system where they will be used. This usually means first installing software packages from the distribution which are needed for the "build" process. Normally, these will be the GNU compiler (GCC), GNU Make (make) and packages containing "header files" for your kernel - and making sure that all system updates are installed and that the system is running the most up-to-date kernel included in the distribution. *The running kernel and*

the header files must be updated to matching versions. We will give some instructions for common distributions. For most of them you will want to start by finding the version name of your kernel using the command `uname -r` in a terminal. They assume that you have not changed too much from the original installation, particularly not installed a different kernel type. If you have then you will need to determine yourself what to set up.

- With Debian and Ubuntu-based distributions, you must install the right version of the `linux-headers`, usually whichever of `linux-headers-generic` , `linux-headers-amd64` , `linux-headers-i686` or `linux-headers-i686-pae` best matches the kernel version name; and if it exists the `linux-kbuild` package. Basic Ubuntu releases should have the right packages installed by default.
- On Fedora, Redhat, Oracle Linux and many other RPM-based systems, the kernel version sometimes has a code of letters or a word close to the end of the version name, for example "uek" for the Oracle Enterprise kernel or "default" or "desktop" for the standard SUSE kernels. In this case the package name is `kernel-uek-devel` or equivalent. If there is no such code, it is usually `kernel-devel`.
- On older SUSE and openSUSE Linux, you must install the `kernel-source` and `kernel-syms` packages.

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

```
rcvboxdrv setup
```

2.3.3 Performing the installation

VirtualBox is available in a number of package formats native to various common Linux distributions (see chapter 1.4, *Supported host operating systems*, page 15 for details). In addition, there is an alternative generic installer (.run) which should work on most Linux distributions. The generic installer packages are built on EL5 systems and thus require reasonable old versions of glibc (version 2.5) and other system libraries.

2.3.3.1 Installing VirtualBox from a Debian/Ubuntu package

First, download the appropriate package for your distribution. The following examples assume that you are installing to a 32-bit Ubuntu Wily system. Use `dpkg` to install the Debian package:

```
sudo dpkg -i virtualbox-5.0_5.1.22_Ubuntu_wily_i386.deb
```

The installer will also try to build kernel modules suitable for the current running kernel. If the build process is not successful you will be shown a warning and the package will be left unconfigured. 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 driver modules*, page 37). After correcting any problems, do

```
sudo rcvboxdrv setup
```

This will start a second attempt to build the module.

If a suitable kernel module was found in the package or the module was successfully built, the installation script will attempt to load that module. If this fails, please see chapter 12.8.1, *Linux kernel module refuses to load*, page 260 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 42).

2.3.3.2 Using the alternative generic installer (VirtualBox.run)

The alternative generic installer performs the following steps:

- It unpacks the application files to the target directory,

  `/opt/VirtualBox/`

  which cannot be changed.
- It builds the VirtualBox kernel modules (`vboxdrv`, `vboxnetflt` and `vboxnetadp`) and installs them.
- It creates `/sbin/rcvboxdrv`, an init script to start the VirtualBox kernel module.
- It creates a new system group called `vboxusers`.
- It creates symbolic links in `/usr/bin` to the a shell script (`/opt/VirtualBox/VBox`) which does some sanity checks and dispatches to the actual executables, `VirtualBox`, `VBoxSDL`, `VBoxVRDP`, `VBoxHeadless` and `VBoxManage`
- It creates `/etc/udev/rules.d/60-vboxdrv.rules`, a description file for udev, if that is present, which makes the USB devices accessible to all users in the `vboxusers` group.
- 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.

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

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

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

After that you need to put every user which should be able to access USB devices from VirtualBox guests 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`.

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

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

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

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

2.3.3.4 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
```

. Starting with version 2.2.2, you can uninstall the .run package by invoking

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

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

2.3.3.5 Automatic installation of Debian packages

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

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

The first line 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.

In addition there are some common configuration options that can be set prior to the installation, described in chapter 2.3.3.7, *Automatic installation options*, page 41.

2.3.3.6 Automatic installation of .rpm packages

The .rpm format does not provide a configuration system comparable to the debconf system. See chapter 2.3.3.7, *Automatic installation options*, page 41 for how to set some common installation options provided by VirtualBox.

2.3.3.7 Automatic installation options

To configure the installation process of our .deb and .rpm packages, you can create a response file named `/etc/default/virtualbox`. 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 fitting the current kernel was found.

2.3.4 The vboxusers group

The Linux installers create the system user group `vboxusers` during installation. Any system user who is going to use USB devices from VirtualBox guests must be a member of that group. A user can be made a member of the group `vboxusers` through the GUI user/group management or at the command line with

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

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 Solaris hosts

For the specific versions of Solaris that we support as host operating systems, please refer to chapter 1.4, *Supported host operating systems*, page 15.

If you have a previously installed instance of VirtualBox on your Solaris host, please uninstall it first before installing a new instance. Refer to chapter 2.4.4, *Uninstallation*, page 43 for uninstall instructions.

2.4.1 Performing the installation

VirtualBox is available as a standard Solaris package. Download the VirtualBox SunOS package which includes the 64-bit versions of VirtualBox. *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-5.1.22-SunOS.tar.gz | tar xvf -
```

Starting with VirtualBox 3.1 the VirtualBox kernel package is no longer a separate package and has been integrated into the main package. Install the VirtualBox package using:

```
pkgadd -d VirtualBox-5.1.22-SunOS.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 2.4.6, *Configuring a zone for running VirtualBox*, page 43.

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 The vboxuser group

Starting with VirtualBox 4.1, the installer creates the system user group `vboxuser` during installation for Solaris hosts that support the USB features required by VirtualBox. Any system user who is going to use USB devices from VirtualBox guests must be a member of this group. A user can be made a member of this group through the GUI user/group management or at the command line by executing as root:

```
usermod -G vboxuser username
```

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.

2.4.3 Starting VirtualBox on Solaris

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 the `VirtualBox` Qt GUI such as fonts and colours by executing `VBoxQtconfig` from the terminal.

2.4.4 Uninstallation

Uninstallation of VirtualBox on Solaris 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.

If you are uninstalling VirtualBox version 3.0 or lower, you need to remove the VirtualBox kernel interface package, execute:

```
pkgrm SUNWvboxkern
```

2.4.5 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-5.1.22-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
```

2.4.6 Configuring a zone for running VirtualBox

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
```

Replace “vboxzone” with the name of the zone in which you intend to run VirtualBox.

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

If you are running VirtualBox 2.2.0 or above on Solaris 11 or above, you may add a device for `/dev/vboxusbmon` too, similar to what was shown above. This does not apply to Solaris 10 hosts due to lack of USB support.

Next reboot the zone using `zoneadm` and you should be able to run VirtualBox from within the configured zone.

3 Configuring virtual machines

Whereas chapter 1, *First steps*, page 11 gave you a quick introduction to VirtualBox and how to get your first virtual machine running, the following chapter describes in detail how to configure virtual machines.

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 so choose, give the host system, other guests, or computers on the Internet access to the guest system.

3.1 Supported guest operating systems

Since VirtualBox is designed to provide a generic virtualization environment for x86 systems, it may run operating systems of any kind, even those not listed here. However, the focus is to optimize VirtualBox for the following guest systems:

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 / Server 2008 / 7 / 8 / 8.1 / 10 RTM 10240 / Server 2012 All versions, editions and service packs are fully supported (including 64-bit versions, under the preconditions listed below). Guest Additions are available. Windows 8 and later requires hardware virtualization to be enabled.

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 (32 bits and 64 bits). Guest Additions are available.

We strongly recommend using a Linux kernel version 2.6.13 or higher for better performance.

> **Note:** Certain Linux kernel releases have bugs that prevent them from executing in a virtual environment; please see chapter 12.4.3, *Buggy Linux 2.6 kernel versions*, page 257 for details.

Linux 3.x All versions/editions are fully supported (32 bits and 64 bits). Guest Additions are available.

Solaris 10 (u6 and higher), Solaris 11 (including Solaris 11 Express) Fully supported (64 bits, prior to Solaris 11 11/11 also 32 bits). Guest Additions are available.

FreeBSD Requires hardware virtualization to be enabled. Limited support. Guest Additions are not available yet.

OpenBSD Requires hardware virtualization to be enabled. Versions 3.7 and later 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.[1]

Mac OS X VirtualBox 3.2 added experimental support for Mac OS X guests, but this comes with restrictions. Please see the following section as well as chapter 14, *Known limitations*, page 269.

3.1.1 Mac OS X guests

Starting with version 3.2, VirtualBox has experimental support for Mac OS X guests. This allows you to install and execute unmodified versions of Mac OS X on supported host hardware.

Whereas competing solutions perform modifications to the Mac OS X install DVDs (e.g. different boot loader and replaced files), VirtualBox is the first product to provide the modern PC architecture expected by OS X without requiring any "hacks".

You should be aware of a number of **important issues** before attempting to install a Mac OS X guest:

1. Mac OS X is commercial, licensed software and contains **both license and technical restrictions** that limit its use to certain hardware and usage scenarios. It is important that you understand and obey these restrictions.

   In particular, for most versions of Mac OS X, Apple prohibits installing them on non-Apple hardware.

   These license restrictions are also enforced on a technical level. Mac OS X verifies whether it is running on Apple hardware, and most DVDs that come with Apple hardware even check for an exact model. These restrictions are *not* circumvented by VirtualBox and continue to apply.

2. Only **CPUs** known and tested by Apple are supported. As a result, if your Intel CPU is newer than the build of Mac OS X, or if you have a non-Intel CPU, it will most likely panic during bootup with an "Unsupported CPU" exception. It is generally best to use the Mac OS X DVD that came with your Apple hardware.

3. The Mac OS X installer expects the harddisk to be **partitioned** so when it does not offer a selection, you have to launch the Disk Utility from the "Tools" menu and partition the hard disk. Then close the Disk Utility and proceed with the installation.

4. In addition, as Mac OS X support in VirtualBox is currently still experimental, please refer also to chapter 14, *Known limitations*, page 269.

3.1.2 64-bit guests

VirtualBox supports 64-bit guest operating systems, even on 32-bit host operating systems,[2] provided that the following conditions are met:

1. You need a 64-bit processor with hardware virtualization support (see chapter 10.3, *Hardware vs. software virtualization*, page 239).

2. You must enable hardware virtualization for the particular VM for which you want 64-bit support; software virtualization is not supported for 64-bit VMs.

[1] See chapter 14, *Known limitations*, page 269.

[2] 64-bit guest support was added with VirtualBox 2.0; support for 64-bit guests on 32-bit hosts was added with VirtualBox 2.1.

3. If you want to use 64-bit guest support on a 32-bit host operating system, you must also select a 64-bit operating system for the particular VM. Since supporting 64 bits on 32-bit hosts incurs additional overhead, VirtualBox only enables this support upon explicit request.

   On 64-bit hosts (which typically come with hardware virtualization support), 64-bit guest operating systems are always supported regardless of settings, so you can simply install a 64-bit operating system in the guest.

Warning: On any host, you should enable the **I/O APIC** for virtual machines that you intend to use in 64-bit mode. This is especially true for 64-bit Windows VMs. See chapter 3.3.2, *"Advanced" tab*, page 48. In addition, for 64-bit Windows guests, you should make sure that the VM uses the **Intel networking device**, since there is no 64-bit driver support for the AMD PCNet card; see chapter 6.1, *Virtual networking hardware*, page 92.

If you use the "Create VM" wizard of the VirtualBox graphical user interface (see chapter 1.7, *Creating your first virtual machine*, page 18), VirtualBox will automatically use the correct settings for each selected 64-bit operating system type.

3.2 Emulated hardware

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

- **Input devices.** By default, VirtualBox emulates a standard PS/2 keyboard and mouse. These devices are supported by almost all past and present operating systems.

  In addition, VirtualBox can provide virtual USB input devices to avoid having to capture mouse and keyboard, as described in chapter 1.8.2, *Capturing and releasing keyboard and mouse*, page 22.

- **Graphics.** The VirtualBox graphics device (sometimes referred to as VGA device) is, unlike nearly all other emulated devices, not based on any physical counterpart. It is a simple, synthetic device which provides compatibility with standard VGA and several extended registers used by the VESA BIOS Extensions (VBE).

- **Storage.** VirtualBox currently emulates the standard ATA interface found on Intel PIIX3/PIIX4 chips, the SATA (AHCI) interface, and two SCSI adapters (LSI Logic and BusLogic); see chapter 5.1, *Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe*, page 79 for details. Whereas providing one of these would be enough for VirtualBox by itself, this multitude of storage adapters is required for compatibility with other hypervisors. Windows is particularly picky about its boot devices, and migrating VMs between hypervisors is very difficult or impossible if the storage controllers are different.

- **Networking.** See chapter 6.1, *Virtual networking hardware*, page 92.

- **USB.** VirtualBox emulates three USB host controllers: xHCI, EHCI, and OHCI. While xHCI handles all USB transfer speeds, only guest operating systems released approximately after 2011 support xHCI. Note that for Windows 7 guests, 3rd party drivers must be installed for xHCI support.

  Older operating systems typically support OHCI and EHCI. The two controllers are needed because OHCI only handles USB low- and full-speed devices (both USB 1.x and 2.0), while EHCI only handles high-speed devices (USB 2.0 only).

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

- **Audio.** See chapter 3.7, *Audio settings*, page 54.

3.3 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 three tabs, "Basic", "Advanced" and "Description".

3.3.1 "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`.

Operating system / version 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 in chapter 1.7, *Creating your first virtual machine*, page 18.

Whereas the default settings of a newly created VM depend on the selected operating system type, changing the type later has no effect on VM settings; this value is then purely informational and decorative.

3.3.2 "Advanced" tab

Snapshot Folder By default, VirtualBox saves snapshot data together with your other VirtualBox configuration data; see chapter 10.1, *Where VirtualBox stores its files*, page 234. With this setting, you can specify any other folder for each VM.

Shared Clipboard 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.

Clipboard sharing requires that the VirtualBox Guest Additions be installed. As a result, this setting has no effect otherwise; see chapter 4, *Guest Additions*, page 60 for details.

The shared clipboard is disabled by default. See chapter 13.3.2.3, *Clipboard*, page 266 for an explanation. This setting can be changed at any time using the "Shared Clipboard" menu item in the "Devices" menu of the virtual machine.

Drag and Drop This setting allows to enable support for drag and drop: Select an object (e.g. a file) from the host or guest and directly copy or open it on the guest or host. Multiple per-VM drag and drop modes allow restricting access in either direction.

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

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

See chapter 4.4, *Drag and Drop*, page 70 for more information. [3]

3.3.3 "Description" tab

Here you can enter any description for your virtual machine, if you want. This has no effect on 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.

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

3.4 System settings

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

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

3.4.1 "Motherboard" tab

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

Base memory This sets the amount of RAM that is allocated and given to the VM when it is running. The specified amount of memory will be requested from the host operating system, so it must be available or made available as free memory on the host when attempting to start the VM and will not be available to the host while the VM is running. This is the same setting that was specified in the "New Virtual Machine" wizard, as described with guidelines under chapter 1.7, *Creating your first virtual machine*, page 18 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).

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), the network, or none of these.

If you select "Network", the VM will attempt to boot from a network via the PXE mechanism. This needs to be configured in detail on the command line; please see chapter 8.8, *VBoxManage modifyvm*, page 128.

Chipset Here you can select which chipset will be presented to the virtual machine. Before VirtualBox 4.0, PIIX3 was the only available option here. For modern guest operating systems such as Mac OS X, that old chipset is no longer well supported. As a result, VirtualBox 4.0 introduced an emulation of the more modern ICH9 chipset, which supports PCI express, three PCI buses, PCI-to-PCI bridges and Message Signaled Interrupts (MSI). This allows modern operating systems to address more PCI devices and no longer requires IRQ sharing. Using the ICH9 chipset it is also possible to configure up to 36 network cards (up to 8 network adapters with PIIX3). Note that the ICH9 support is experimental and not recommended for guest operating systems which do not require it.

[3] Experimental support for drag and drop was added with VirtualBox 4.2.

Pointing Device The default virtual pointing devices for older guests is the traditional PS/2 mouse. If set to *USB tablet*, VirtualBox reports to the virtual machine that a USB tablet device is present and communicates mouse events to the virtual machine through this device. The third setting is a *USB Multi-Touch Tablet* which is suited for recent Windows guests.

Using the virtual USB tablet has the advantage that movements are reported in absolute coordinates (instead of as relative position changes), which allows VirtualBox to translate mouse events over the VM window into tablet events without having to "capture" the mouse in the guest as described in chapter 1.8.2, *Capturing and releasing keyboard and mouse*, page 22. This makes using the VM less tedious even if Guest Additions are not installed.[4]

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.

> **Note:** Enabling the I/O APIC is *required* for 64-bit guest operating systems, especially Windows Vista; it is also required if you want to use more than one virtual CPU in a virtual machine.

However, software support for I/O APICs has been unreliable with some 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 EFI This enables Extensible Firmware Interface (EFI), which replaces the legacy BIOS and may be useful for certain advanced use cases. Please refer to chapter 3.13, *Alternative firmware (EFI)*, page 59 for details.

Hardware clock in UTC time If checked, VirtualBox will report the system time in UTC format to the guest instead of local (host) time. This affects how the virtual real-time clock (RTC) operates and may be useful for Unix-like guest operating systems, which typically expect the hardware clock to be set to UTC.

In addition, you can turn off the **Advanced Configuration and Power Interface (ACPI)** which VirtualBox presents to the guest operating system by default. 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. It can only be turned off on the command line; see chapter 8.8, *VBoxManage modifyvm*, page 128.

> **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 of a Windows guest OS. Turning it on after installation will have no effect however.

[4] The virtual USB tablet was added with VirtualBox 3.2. Depending on the guest operating system selected, this is now enabled by default for new virtual machines.

3.4.2 "Processor" tab

On the "Processor" tab, you can set how many virtual **CPU cores** the guest operating systems should see. Starting with version 3.0, VirtualBox supports symmetrical multiprocessing (SMP) and can present up to 32 virtual CPU cores to each virtual machine.

You should not, however, configure virtual machines to use more CPU cores than you have available physically (real cores, no hyperthreads).

On this tab you can also set the **"CPU execution cap"**. This setting limits the amount of time a host CPU spends to emulate a virtual CPU. The default setting is 100% meaning that there is no limitation. A setting of 50% implies a single virtual CPU can use up to 50% of a single host CPU. Note that limiting the execution time of the virtual CPUs may induce guest timing problems.

In addition, the **"Enable PAE/NX"** 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.

With virtual machines running modern server operating systems, VirtualBox also supports CPU hot-plugging. For details about this, please refer to chapter 9.5, *CPU hot-plugging*, page 193.

3.4.3 "Acceleration" tab

On this page, you can determine whether and how VirtualBox should use hardware virtualization extensions that your host CPU may support. This is the case with most CPUs built after 2006.

You can select for each virtual machine individually whether VirtualBox should use software or hardware virtualization.[5]

In most cases, the default settings will be fine; VirtualBox will have picked sensible defaults depending on the operating system that you selected when you created the virtual machine. In certain situations, however, you may want to change these preconfigured defaults.

Advanced users may be interested in technical details about software vs. hardware virtualization; please see chapter 10.3, *Hardware vs. software virtualization*, page 239.

If your host's CPU supports the **nested paging** (AMD-V) or **EPT** (Intel VT-x) features, then you can expect a significant performance increase by enabling nested paging in addition to hardware virtualization. For technical details, see chapter 10.7, *Nested paging and VPIDs*, page 243.

Starting with version 5.0, VirtualBox provides paravirtualization interfaces to improve timekeeping accuracy and performance of guest operating systems. The options available are documented under the `paravirtprovider` option in chapter 8.8, *VBoxManage modifyvm*, page 128. For further details on the paravirtualization providers, please refer to chapter 10.4, *Paravirtualization providers*, page 240.

3.5 Display settings

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

The GUI will show a warning if the amount of video memory is too small to be able to switch the VM into full screen mode. The minimum value depends on the number of virtual monitors, the screen resolution and the color depth of the host display as well as

[5] Prior to VirtualBox version 2.2, software virtualization was the default; starting with version 2.2, VirtualBox will enable hardware virtualization by default for new virtual machines that you create. (Existing virtual machines are not automatically changed for compatibility reasons, and the default can of course be changed for each virtual machine.)

of the activation of *3D acceleration* and *2D video acceleration*. A rough estimate is (*color depth* / 8) x *vertical pixels* x *horizontal pixels* x *number of screens* = *number of bytes*. Like said above, there might be extra memory required for any activated display acceleration setting.

Monitor count With this setting VirtualBox can provide more than one virtual monitor to a virtual machine. If a guest operating system (such as Windows) supports multiple attached monitors, VirtualBox can pretend that multiple virtual monitors are present.[6] Up to 8 such virtual monitors are supported.

The output of the multiple monitors will be displayed on the host in multiple VM windows which are running side by side.

However, in full screen and seamless mode, they will use the available physical monitors attached to the host. As a result, for full screen and seamless modes to work with multiple monitors, you will need at least as many physical monitors as you have virtual monitors configured, or VirtualBox will report an error. You can configure the relationship between guest and host monitors using the view menu by pressing Host key + Home when you are in full screen or seamless mode.

Please see chapter 14, *Known limitations*, page 269 also.

Enable 3D acceleration If a virtual machine has Guest Additions installed, you can select here whether the guest should support accelerated 3D graphics. Please refer to chapter 4.5.1, *Hardware 3D acceleration (OpenGL and Direct3D 8/9)*, page 72 for details.

Enable 2D video acceleration If a virtual machine with Microsoft Windows has Guest Additions installed, you can select here whether the guest should support accelerated 2D video graphics. Please refer to chapter 4.5.2, *Hardware 2D video acceleration for Windows guests*, page 73 for details.

Remote display Under the "Remote display" tab, if the VirtualBox Remote Display Extension (VRDE) is installed, you can enable the VRDP server that is built into VirtualBox. This allows you to connect to the console of the virtual machine remotely with any standard RDP viewer, such as `mstsc.exe` that comes with Microsoft Windows. On Linux and Solaris systems you can use the standard open-source `rdesktop` program. These features are described in detail in chapter 7.1, *Remote display (VRDP support)*, page 104.

Video Capture Under the "Video Capture" tab you can enable video capturing for this VM. Note that this feature can also be enabled/disabled while the VM is executed.

3.6 Storage settings

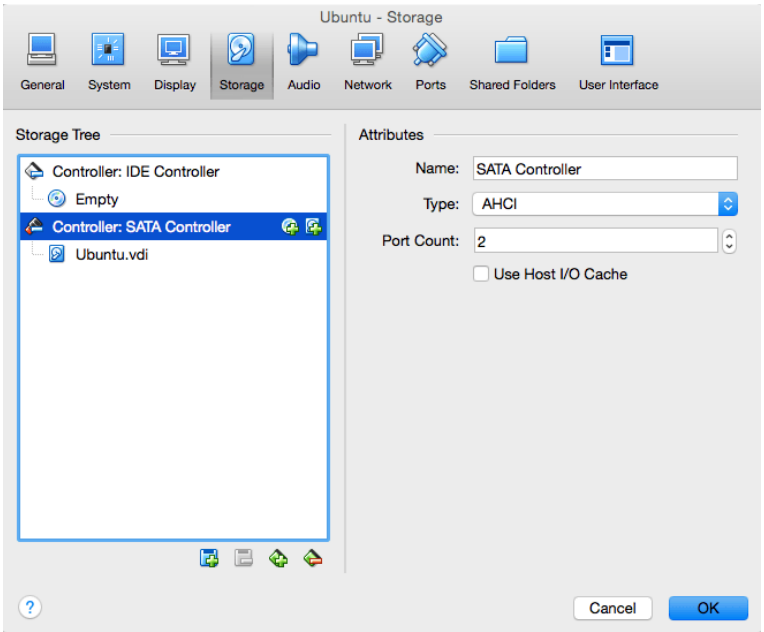
The "Storage" category in the VM settings allows you to connect virtual hard disk, CD/DVD and floppy images and drives to your virtual machine.

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

> **Note:** This section can only give you a quick introduction to the VirtualBox storage settings. Since VirtualBox gives you an enormous wealth of options in this area, we have dedicated an entire chapter of this User Manual to explaining all the details: please see chapter 5, *Virtual storage*, page 79.

[6] Multiple monitor support was added with VirtualBox 3.2.

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



Depending on the guest operating system type that you selected when you created the VM, the typical layout of storage devices in a new VM is as follows:

- You will see an **IDE controller,** to which a virtual CD/DVD drive has been attached (to the “secondary master” port of the IDE controller).
- You will also see a **SATA controller,** which is a more modern type of storage controller for higher hard disk data throughput, to which the virtual hard disks are attached. Initially you will normally have one such virtual disk, but as you can see in the above screenshot, you can have more than one, each represented by a disk image file (VDI files, in this case).

If you created your VM with an older version of VirtualBox, the default storage layout may differ. You might then only have an IDE controller to which both the CD/DVD drive and the hard disks have been attached. This might also apply if you selected an older operating system type when you created the VM. Since older operating systems do not support SATA without additional drivers, VirtualBox will make sure that no such devices are present initially. Please see chapter 5.1, *Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe*, page 79 for additional information.

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

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

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

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

1. You can then select to which **device slot** of the controller the virtual disk should be connected to. IDE controllers have four slots which have traditionally been called "primary master", "primary slave", "secondary master" and "secondary slave". By contrast, SATA and SCSI controllers offer you up to 30 slots to which virtual devices can be attached.
2. You can select which **image file** to use.
   - For virtual hard disks, a button with a drop-down list appears on the right, offering you to either select a **virtual hard disk file** using a standard file dialog or to **create a new hard disk** (image file), which will bring up the "Create new disk" wizard, which was described in chapter 1.7, *Creating your first virtual machine*, page 18.

     For details on the image file types that are supported, please see chapter 5.2, *Disk image files (VDI, VMDK, VHD, HDD)*, page 82.
   - For virtual CD/DVD drives, the image files will typically be in the standard ISO format instead. Most commonly, you will select this option when installing an operating system from an ISO file that you have obtained from the Internet. For example, most Linux distributions are available in this way.

     For virtual CD/DVD drives, the following additional options are available:
     * If you select **"Host drive"** from the list, then the physical device of the host computer is connected to the VM, so that the guest 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.

       If you want to write (burn) CDs or DVDs using the host drive, you need to also enable the **"Passthrough"** option; see chapter 5.9, *CD/DVD support*, page 90.
     * If you select **"Remove disk from virtual drive",** VirtualBox will present an empty CD/DVD drive to the guest into which no media has been inserted.

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

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

3.7 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, an Intel HD Audio controller[7] or a SoundBlaster 16 card. In any case, you can select what audio driver VirtualBox will use on the host.

On a Linux host, depending on your host configuration, you can also select between the OSS, ALSA or the PulseAudio subsystem. On newer Linux distributions, the PulseAudio subsystem should be preferred.

Since VirtualBox 5.0 only OSS is supported on Solaris hosts - the "Solaris Audio" audio backend is no longer supported on Solaris hosts.

[7] Intel HD Audio support was added with VirtualBox 4.0 because Windows 7 and later (as well as 64-bit Windows Vista) do not support the Intel AC'97 controller out of the box.

3.8 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 virtual network card and selects the "Network Address Translation" (NAT) mode for it. This way the guest can connect to the outside world using the host's networking and the outside world can connect to services on the guest which you choose to make visible outside of the virtual machine.

This default setup is good for probably 95% of VirtualBox users. However, VirtualBox is extremely flexible in how it can virtualize networking. It supports many virtual network cards per virtual machine, the first four of which can be configured in detail in the Manager window. Additional network cards can be configured on the command line with VBoxManage.

Because of the vast array of options available, we have dedicated an entire chapter of this manual to discussing networking configuration; please see chapter 6, *Virtual networking*, page 92.

3.9 Serial ports

VirtualBox fully supports virtual serial ports in a virtual machine in an easy-to-use manner.[8]

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

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

If a virtual serial port is enabled, the guest operating system sees a standard 16550A compatible UART device. Both receiving and transmitting data is supported. How this virtual serial port is then connected to the host is configurable, and the details depend on your host 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 8.8, *VBoxManage modifyvm*, page 128; in that section, look for the `--uart` and `--uartmode` options.

In either case, you can configure up to four virtual serial ports per virtual machine. 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[9], 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. Then, you will need to determine what this virtual port should be connected to. For each virtual serial port, you have the following options:

[8] Serial port support was added with VirtualBox 1.5.
[9] See, for example, `http://en.wikipedia.org/wiki/COM_(hardware_interface)`.

- You can elect to have the virtual serial port "disconnected", which means that the guest will see the device, 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 Solaris 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. The pipe name must be in the format `\\.\pipe\<name>` where `<name>` should identify the virtual machine but may be freely chosen.
  - On a Mac, Linux or Solaris host, a local domain socket is used instead. The socket filename must be chosen such that the user running VirtualBox has sufficient privileges to create and write to it. The `/tmp` directory is often a good candidate.

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

  In this case, you can configure whether 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.

  For a direct connection between two virtual machines (corresponding to a null-modem cable), simply configure one VM to create a pipe/socket and another to attach to it.
- You can send the virtual serial port output to a file. This option is very useful for capturing diagnostic output from a guest. Any file may be used for this purpose, as long as the user running VirtualBox has sufficient privileges to create and write to the file.
- TCP Socket: Useful for forwarding serial traffic over TCP/IP, acting as a server, or it can act as a TCP client connecting to other servers. It allows a remote machine to directly connect to the guest's serial port via TCP.
  - TCP Server: Uncheck the *Connect to existing pipe/socket* checkbox and specify the **`port`** number. Typically 23 or 2023. Note that on UNIX-like systems you will have to use a port a number greater than 1024 for regular users.

    The client can use software such as `PuTTY` or the `telnet` command line tool to access the TCP Server.
  - TCP Client: To create a virtual null-modem cable over the Internet or LAN, the other side can connect via TCP by specifying **`hostname:port`**. The TCP socket will act in client mode if check the *Connect to existing pipe/socket* checkbox.

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

3.10 USB support

3.10.1 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 the guest operating system with a virtual USB controller. As soon as the guest system starts using a USB device, it will appear as unavailable on the host.

Note:

1. 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.

2. Solaris hosts have a few known limitations regarding USB support; please see chapter 14, *Known limitations*, page 269.

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 VirtualBox Remote Desktop Extension (VRDE). For details about this, see chapter 7.1.4, *Remote USB*, page 108.

In the Settings dialog, you can first configure whether USB is available in the guest at all, and then choose the level of USB support: OHCI for USB 1.1, EHCI (which will also enable OHCI) for USB 2.0, or xHCI for all USB speeds.

Note: The xHCI and EHCI controllers are shipped as a VirtualBox extension package, which must be installed separately. See chapter 1.5, *Installing VirtualBox and extension packs*, page 16 for more information.

When USB support is enabled for a VM, you can determine in detail which devices will be automatically attached to the guest. For this, you can create so-called "filters" by specifying certain properties of the USB device. USB devices with a matching filter will be automatically passed to the guest once they are attached to the host. USB devices without a matching filter can be passed manually to the guest, for example by using the Devices / USB devices menu.

Clicking on the "+" button to the right of the "USB Device Filters" window creates a **new filter.** You can give the filter a name (for 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*, page 114):

   ```
   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.

3.10.2 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 newer Linux hosts, VirtualBox accesses USB devices through special files in the file system. When VirtualBox is installed, these are made available to all users in the `vboxusers` system group. In order to be able to access USB from guest systems, make sure that you are a member of this group.

On older Linux hosts, USB devices are accessed using 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. The `Driver=` entry in `/proc/bus/usb/devices` will show you which devices are currently claimed. Please refer to chapter 12.8.7, *USB not working*, page 262 also for details about `usbfs`.

3.11 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 is described in detail in chapter 4.3, *Shared folders*, page 68.

3.12 User Interface

The "User Interface" section allows you to change certain aspects of the user interface of this VM.

Menu Bar This widget allows you to disable certain menus (click at the menu to release it), certain menu entries (uncheck the checkbox of the entry to disable it) and the complete menu bar (uncheck the rightmost checkbox).

Mini ToolBar In full screen or seamless mode, VirtualBox can display a small toolbar that contains some of the items that are normally available from the virtual machine's menu bar. This toolbar reduces itself to a small gray line unless you move the mouse over it. With the

toolbar, you can return from full screen or seamless mode, control machine execution or enable certain devices. If you don't want to see the toolbar, disable this setting.

The second setting allows to show the toolbar at the top of the screen instead of showing it at the bottom.

Status Bar This widget allows you to disable certain icons of the status bar (uncheck the checkbox of an icon to disable it), to re-arrange icons (drag and drop the icon) and to disable the complete status bar (uncheck the leftmost checkbox).

3.13 Alternative firmware (EFI)

Starting with release 3.1, VirtualBox includes experimental support for the Extensible Firmware Interface (EFI), which is a new industry standard intended to eventually replace the legacy BIOS as the primary interface for bootstrapping computers and certain system services later.

By default, VirtualBox uses the BIOS firmware for virtual machines. To use EFI for a given virtual machine, you can enable EFI in the machine's "Settings" dialog (see chapter 3.4.1, *"Motherboard" tab*, page 49). Alternatively, use the `VBoxManage` command line interface like this:

```
VBoxManage modifyvm "VM name" --firmware efi
```

To switch back to using the BIOS, use:

```
VBoxManage modifyvm "VM name" --firmware bios
```

One notable user of EFI is Apple's Mac OS X, but more recent Linuxes and Windows (starting with Vista) offer special versions that can be booted using EFI as well.

Another possible use of EFI in VirtualBox is development and testing of EFI applications, without booting any OS.

Note that the VirtualBox EFI support is experimental and will be enhanced as EFI matures and becomes more widespread. Mac OS X, Linux and newer Windows guests are known to work fine. Windows 7 guests are unable to boot with the VirtualBox EFI implementation.

3.13.1 Video modes in EFI

EFI provides two distinct video interfaces: GOP (Graphics Output Protocol) and UGA (Universal Graphics Adapter). Mac OS X uses GOP, while Linux tends to use UGA. VirtualBox provides a configuration option to control the framebuffer size for both interfaces.

To control GOP, use the following `VBoxManage` command:

```
VBoxManage setextradata "VM name" VBoxInternal2/EfiGopMode N
```

Where N can be one of 0,1,2,3,4,5 referring to the 640x480, 800x600, 1024x768, 1280x1024, 1440x900, 1920x1200 screen resolution respectively.

To change the UGA resolution:

```
VBoxManage setextradata "VM name" VBoxInternal2/UgaHorizontalResolution 1440
VBoxManage setextradata "VM name" VBoxInternal2/UgaVerticalResolution    900
```

The video mode for both GOP and UGA can only be changed when the VM is powered off and remains persistent until changed.

3.13.2 Specifying boot arguments

It is currently not possible to manipulate EFI variables from within a running guest (e.g., setting the "boot-args" variable by running the `nvram` tool in a Mac OS X guest will not work). As an alternative way, "VBoxInternal2/EfiBootArgs" extradata can be passed to a VM in order to set the "boot-args" variable. To change the "boot-args" EFI variable:

```
VBoxManage setextradata "VM name" VBoxInternal2/EfiBootArgs <value>
```

4 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 mentioned in chapter 1.2, *Some terminology*, page 12, the Guest Additions are designed to be installed *inside* a virtual machine after the guest operating system has been installed. They consist of device drivers and system applications that optimize the guest operating system for better performance and usability. Please see chapter 3.1, *Supported guest operating systems*, page 45 for details on what guest operating systems are fully supported with Guest Additions by VirtualBox.

The VirtualBox Guest Additions for all supported guest operating systems are provided as a single CD-ROM image file which is called `VBoxGuestAdditions.iso`. This image file is located in the installation directory of VirtualBox. To install the Guest Additions for a particular VM, you mount this ISO file in your VM as a virtual CD-ROM and install from there.

The Guest Additions offer the following features:

Mouse pointer integration To overcome the limitations for mouse support that were described in chapter 1.8.2, *Capturing and releasing keyboard and mouse*, page 22, this provides you with seamless mouse support. You will only have one mouse pointer and pressing the Host key is no longer required to "free" the mouse from being captured by the guest OS. To make this work, a special mouse driver is installed in the guest that communicates with the "real" mouse driver on your host and moves the guest mouse pointer accordingly.

Shared folders These provide an easy way to exchange files between the host and the guest. Much like ordinary Windows network shares, you can tell VirtualBox to treat a certain host directory as a shared folder, and VirtualBox will make it available to the guest operating system as a network share, irrespective of whether guest actually has a network. For details, please refer to chapter 4.3, *Shared folders*, page 68.

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, Linux and Solaris guests, you can resize the virtual machine's window if the Guest Additions are installed. The video resolution in the guest will be automatically adjusted (as if you had manually entered an arbitrary resolution in the guest's display settings). Please see chapter 1.8.5, *Resizing the machine's window*, page 24 also.

Finally, if the Guest Additions are installed, 3D graphics and 2D video for guest applications can be accelerated; see chapter 4.5, *Hardware-accelerated graphics*, page 72.

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 73 for details.

Generic host/guest communication channels The Guest Additions enable you to control and monitor guest execution in ways other than those mentioned above. The so-called "guest properties" provide a generic string-based mechanism to exchange data bits between a guest and a host, some of which have special meanings for controlling and monitoring the guest; see chapter 4.7, *Guest properties*, page 74 for details.

Additionally, applications can be started in a guest from the host; see chapter 4.8, *Guest control*, page 76.

Time synchronization With the Guest Additions installed, VirtualBox can ensure that the guest's system time is better synchronized with that of the host.

For various reasons, the time in the guest might run at a slightly different rate than the time on the host. The host could be receiving updates via NTP and its own time might not run linearly. A VM could also be paused, which stops the flow of time in the guest for a shorter or longer period of time. When the wall clock time between the guest and host only differs slightly, the time synchronization service attempts to gradually and smoothly adjust the guest time in small increments to either "catch up" or "lose" time. When the difference is too great (e.g., a VM paused for hours or restored from saved state), the guest time is changed immediately, without a gradual adjustment.

The Guest Additions will re-synchronize the time regularly. See chapter 9.14.3, *Tuning the Guest Additions time synchronization parameters*, page 206 for how to configure the parameters of the time synchronization mechanism.

Shared clipboard With the Guest Additions installed, the clipboard of the guest operating system can optionally be shared with your host operating system; see chapter 3.3, *General settings*, page 48.

Automated logons (credentials passing) For details, please see chapter 9.2, *Automated guest logons*, page 187.

Each version of VirtualBox, even minor releases, ship with their own version of the Guest Additions. While the interfaces through which the VirtualBox core communicates with the Guest Additions are kept stable so that Guest Additions already installed in a VM should continue to work when VirtualBox is upgraded on the host, for best results, it is recommended to keep the Guest Additions at the same version.

Starting with VirtualBox 3.1, the Windows and Linux Guest Additions therefore check automatically whether they have to be updated. If the host is running a newer VirtualBox version than the Guest Additions, a notification with further instructions is displayed in the guest.

To disable this update check for the Guest Additions of a given virtual machine, set the value of its `/VirtualBox/GuestAdd/CheckHostVersion` guest property to `0`; see chapter 4.7, *Guest properties*, page 74 for details.

4.2 Installing and Maintaining Guest Additions

Guest Additions are available for virtual machines running Windows, Linux, Solaris or OS/2. The following sections describe the specifics of each variant in detail.

4.2.1 Guest Additions for Windows

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 Server 2008
- Microsoft Windows Vista (all editions)
- Microsoft Windows 7 (all editions)
- Microsoft Windows 8 (all editions)
- Microsoft Windows 10 RTM build 10240
- Microsoft Windows Server 2012

4.2.1.1 Installation

In the "Devices" menu in the virtual machine's menu bar, VirtualBox has a handy menu item named "Insert Guest Additions CD image", which mounts the Guest Additions ISO file inside your virtual machine. A Windows guest should then automatically start the Guest Additions installer, which installs the Guest Additions into your Windows guest. Other guest operating systems (or if automatic start of software on CD is disabled) need manual start of the installer.

> **Note:** For the basic Direct3D acceleration to work in a Windows Guest, you have to install the Guest Additions in "Safe Mode". This does **not** apply to the experimental WDDM Direct3D video driver available for Vista and Windows 7 guests, see chapter 14, *Known limitations*, page 269 for details.[a]
>
> [a] The experimental WDDM driver was added with VirtualBox 4.1.

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 Media Manager described in chapter 5.3, *The Virtual Media Manager*, page 83.
3. In the Virtual Media 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\Oracle\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.)
   - On a Linux host, you can find this file in the `additions` folder under where you installed VirtualBox (normally `/opt/VirtualBox/`).
   - On Solaris hosts, you can find this file in the `additions` folder under where you installed VirtualBox (normally `/opt/VirtualBox`).
4. Back in the Virtual Media 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.

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 `VBoxWindowsAdditions.exe` from the CD/DVD drive inside the guest to start the installer.

The installer will add several device drivers to the Windows driver database and then invoke the hardware detection wizard.

Depending on your configuration, it might display warnings that the drivers are not digitally signed. You must confirm these in order to continue the installation and properly install the Additions.

After installation, reboot your guest operating system to activate the Additions.

4.2.1.2 Updating the Windows Guest Additions

Windows Guest Additions can be updated by running the installation program again, 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.

4.2.1.3 Unattended Installation

As a prerequisite for avoid popups while performing an unattended installation of the VirtualBox Guest Additions, the code signing certificates used to sign the drivers needs to be installed in the right certificates stores in the guest system. Failing to do this will cause a typical windows installation to pop up a dialog asking whether its allowable to install each driver.

> **Note:** On some Windows versions like Windows 2000 and Windows XP the user intervention popups mentioned above always will be displayed, even after importing the Oracle certificates.

Since VirtualBox 4.2 installing those code signing certificates on a Windows guest can be done in an automated fashion using the `VBoxCertUtil.exe` utility found on the Guest Additions installation CD in the `cert` folder:

- Log in as Administrator on the guest.
- Mount the VirtualBox Guest Additions .ISO.
- Open a command line window on the guest and change to the `cert` folder on the VirtualBox Guest Additions CD.
- Do

  ```
  for %i in (vbox*.cer) do VBoxCertUtil add-trusted-publisher %i --root %i
  ```

  This will install the certificates to the certificate store. When installing the same certificate more than once, an appropriate error will be displayed.

Prior to VirtualBox 4.2 the code signing certificates need to be imported in more manual style using the `certutil.exe` utility, which is shipped since Windows Vista. For Windows versions before Vista you need to download and install `certutil.exe` manually. Since the certificates are not accompanied on the VirtualBox Guest Additions CD-ROM prior to 4.2, these need to get extracted from a signed VirtualBox executable first.

In the following example the needed certificates will be extracted from the VirtualBox Windows Guest Additions installer on the CD-ROM:

VeriSign Code Signing CA Open the Windows Explorer.

- Right click on VBoxWindowsAdditions-<Architecture>.exe, click on "Properties"
- Go to tab "Digital Signatures", choose "Oracle Corporation" and click on "Details"
- In tab "General" click on "View Certificate"
- In tab "Certification Path" select "VeriSign Class 3 Public Primary CA"
- Click on "View Certificate"
- In tab "Details" click on "Copy to File ..."
- In the upcoming wizard choose "DER encoded binary X.509 (.CER)" and save the certificate file to a local path, finish the wizard
- Close certificate dialog for "Verisign Class 3 Code Signing 2010 CA"

Oracle Corporation Open the Windows Explorer.

- Right click on VBoxWindowsAdditions-<Architecture>.exe, click on "Properties"
- Go to tab "Digital Signatures", choose "Oracle Corporation" and click on "Details"
- In tab "General" click on "View Certificate"
- In tab "Details" click on "Copy to File ..."
- In the upcoming wizard choose "DER encoded binary X.509 (.CER)" and save the certificate file to a local path, finish the wizard
- Close certificate dialog for "Oracle Corporation"

After exporting the two certificates above they can be imported into the certificate store using the `certutil.exe` utility:

```
certutil -addstore -f Root "<Path to exported certificate file>"
```

In order to allow for completely unattended guest installations, you can specify a command line parameter to the install launcher:

```
VBoxWindowsAdditions.exe /S
```

This automatically installs the right files and drivers for the corresponding platform (32- or 64-bit).

> **Note:** By default on an unattended installation on a Windows 7 or 8 guest, there will be the XPDM graphics driver installed. This graphics driver does not support Windows Aero / Direct3D on the guest - instead the experimental WDDM graphics driver needs to be installed. To select this driver by default, add the command line parameter `/with_wddm` when invoking the Windows Guest Additions installer.

> **Note:** For Windows Aero to run correctly on a guest, the guest's VRAM size needs to be configured to at least 128 MB.

For more options regarding unattended guest installations, consult the command line help by using the command:

```
VBoxWindowsAdditions.exe /?
```

4.2.1.4 Manual file extraction

If you would like to install the files and drivers manually, you can extract the files from the Windows Guest Additions setup by typing:

```
VBoxWindowsAdditions.exe /extract
```

To explicitly extract the Windows Guest Additions for another platform than the current running one (e.g. 64-bit files on a 32-bit system), you have to execute the appropriate platform installer (`VBoxWindowsAdditions-x86.exe` or `VBoxWindowsAdditions-amd64.exe`) with the `/extract` parameter.

4.2.2 Guest Additions for Linux

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

The following Linux distributions are officially supported:

- Oracle Linux as of version 5 including UEK kernels;
- Fedora as of Fedora Core 4;
- Redhat Enterprise Linux as of version 3;
- SUSE and openSUSE Linux as of version 9;
- Ubuntu as of version 5.10.

Many other distributions are known to work with the Guest Additions.

The version of the Linux kernel supplied by default in SUSE and openSUSE 10.2, Ubuntu 6.10 (all versions) and Ubuntu 6.06 (server edition) contains a bug which can cause it to crash during startup when it is run in a virtual machine. The Guest Additions work in those distributions.

Note that some Linux distributions already come with all or part of the VirtualBox Guest Additions. You may choose to keep the distribution's version of the Guest Additions but these are often not up to date and limited in functionality, so we recommend replacing them with the Guest Additions that come with VirtualBox. The VirtualBox Linux Guest Additions installer tries to detect existing installation and replace them but depending on how the distribution integrates the Guest Additions, this may require some manual interaction. It is highly recommended to take a snapshot of the virtual machine before replacing pre-installed Guest Additions.

4.2.2.1 Installing the Linux Guest Additions

The VirtualBox Guest Additions for Linux are provided on the same virtual CD-ROM file as the Guest 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 generally involves the following steps:

1. Before installing the Guest Additions, you will have to prepare your guest system for building external kernel modules. This works similarly as described in chapter 2.3.2, *The VirtualBox driver modules*, page 37, except that this step must now be performed in your Linux *guest* instead of on a Linux host system, as described there.

   If you suspect that something has gone wrong, check that your guest is set up correctly and try executing the command

   ```
   rcvboxadd setup
   ```

   as root.

2. Insert the `VBoxGuestAdditions.iso` CD file into your Linux guest's virtual CD-ROM drive, exactly the same way as described for a Windows guest in chapter 4.2.1.1, *Installation*, page 62.

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

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

4.2.2.2 Graphics and mouse integration

In Linux and Solaris guests, VirtualBox graphics and mouse integration goes through the X Window System. VirtualBox can use the X.Org variant of the system (or XFree86 version 4.3 which is identical to the first X.Org release). During the installation process, the X.Org display server will be set up to use the graphics and mouse drivers which come with the Guest Additions.

After installing the Guest Additions into a fresh installation of a supported Linux distribution or Solaris system (many unsupported systems will work correctly too), the guest's graphics mode will change to fit the size of the VirtualBox window on the host when it is resized. You can also ask the guest system to switch to a particular resolution by sending a "video mode hint" using the `VBoxManage` tool.

Multiple guest monitors are supported in guests using the X.Org server version 1.3 (which is part of release 7.3 of the X Window System version 11) or a later version. The layout of the guest screens can be adjusted as needed using the tools which come with the guest operating system.

If you want to understand more about the details of how the X.Org drivers are set up (in particular if you wish to use them in a setting which our installer doesn't handle correctly), you should read chapter 9.4.2, *Guest graphics and mouse driver setup in depth*, page 192.

4.2.2.3 Updating the Linux Guest Additions

The Guest Additions can simply be updated by going through the installation procedure again with an updated CD-ROM image. This will replace the drivers with updated versions. You should reboot after updating the Guest Additions.

4.2.2.4 Uninstalling the Linux Guest Additions

If you have a version of the Guest Additions installed on your virtual machine and wish to remove it without installing new ones, you can do so by inserting the Guest Additions CD image into the virtual CD-ROM drive as described above and running the installer for the current Guest Additions with the "uninstall" parameter from the path that the CD image is mounted on in the guest:

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

While this will normally work without issues, you may need to do some manual cleanup of the guest (particularly of the XFree86Config or xorg.conf file) in some cases, particularly if the Additions version installed or the guest operating system were very old, or if you made your own changes to the Guest Additions setup after you installed them.

Starting with version 3.1.0, you can uninstall the Additions by invoking

```
/opt/VBoxGuestAdditions-5.1.22/uninstall.sh
```

Please replace `/opt/VBoxGuestAdditions-5.1.22` with the correct Guest Additions installation directory.

4.2.3 Guest Additions for Solaris

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:

- Solaris 11 including Solaris 11 Express;
- Solaris 10 (u5 and higher);

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

4.2.3.1 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, *Installation*, page 62.

   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 -G -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.

4.2.3.2 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
```

4.2.3.3 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.

4.2.4 Guest Additions for OS/2

VirtualBox also ships with a set of drivers that improve running OS/2 in a virtual machine. Due to restrictions of OS/2 itself, this variant of the Guest Additions has a limited feature set; see chapter 14, *Known limitations*, page 269 for details.

The OS/2 Guest Additions are provided on the same ISO CD-ROM as those for the other platforms. As a result, mount the ISO in OS/2 as described previously. The OS/2 Guest Additions are located in the directory `\32bit\OS2`.

As we do not provide an automatic installer at this time, please refer to the `readme.txt` file in that directory, which describes how to install the OS/2 Guest Additions manually.

4.3 Shared folders

With the "shared folders" feature of VirtualBox, you can access files of your host system from within the guest system. This is similar how you would use network shares in Windows networks – except that shared folders do not need require networking, only the Guest Additions. Shared Folders are supported with Windows (2000 or newer), Linux and Solaris guests.

Shared folders must physically reside on the *host* and are then shared with the guest, which uses a special file system driver in the Guest Addition to talk to the host. For Windows guests, shared folders are implemented as a pseudo-network redirector; for Linux and Solaris guests, the Guest Additions provide a virtual file system.

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 window of a running VM, you can select "Shared folders" from the "Devices" menu, or click on the folder icon on the status bar in the bottom right corner.
- If a VM is not currently running, you can configure shared folders in each virtual machine's "Settings" dialog.
- From the command line, you can create shared folders using VBoxManage, as follows:

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

  See chapter 8.32, *VBoxManage sharedfolder add/remove*, page 163 for details.

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.

Starting with version 4.0, VirtualBox shared folders also support symbolic links (**symlinks**), under the following conditions:

1. The host operating system must support symlinks (i.e. a Mac, Linux or Solaris host is required).
2. Currently only Linux and Solaris Guest Additions support symlinks.
3. For security reasons the guest OS is not allowed to create symlinks by default. If you trust the guest OS to not abuse the functionality, you can enable creation of symlinks for "sharename" with:

   ```
   VBoxManage setextradata "VM name" VBoxInternal2/SharedFoldersEnableSymlinksCreate/sharename 1
   ```

4.3.1 Manual mounting

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, shared folders are browseable and 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
  ```

  To mount a shared folder during boot, add the following entry to /etc/fstab:

  ```
  sharename   mountpoint   vboxsf   defaults  0   0
  ```

- In a Solaris guest, use the following command:

  ```
  mount -F vboxfs [-o OPTIONS] sharename mountpoint
  ```

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

  Here is an example of mounting the shared folder for the user "jack" on Solaris:

  ```
  $ id
  uid=5000(jack) gid=1(other)
  $ mkdir /export/home/jack/mount
  $ pfexec mount -F vboxfs -o uid=5000,gid=1 jackshare /export/home/jack/mount
  $ cd ~/mount
  $ ls
  sharedfile1.mp3 sharedfile2.txt
  $
  ```

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

  ```
  iocharset CHARSET
  ```

  to set the character set used for I/O operations. Note that on Linux guests, if the "iocharset" option is not specified then the Guest Additions driver will attempt to use the character set specified by the CONFIG_NLS_DEFAULT kernel option. If this option is not set either then UTF-8 will be used. Also,

  ```
  convertcp CHARSET
  ```

  is available in order 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.

4.3.2 Automatic mounting

Starting with version 4.0, VirtualBox can mount shared folders automatically, at your option. If automatic mounting is enabled for a specific shared folder, the Guest Additions will automatically mount that folder as soon as a user logs into the guest OS. The details depend on the guest OS type:

- With **Windows guests,** any auto-mounted shared folder will receive its own drive letter (e.g. `E:`) depending on the free drive letters remaining in the guest.

  If there no free drive letters left, auto-mounting will fail; as a result, the number of auto-mounted shared folders is typically limited to 22 or less with Windows guests.

- With **Linux guests,** auto-mounted shared folders are mounted into the `/media` directory, along with the prefix `sf_`. For example, the shared folder `myfiles` would be mounted to `/media/sf_myfiles` on Linux and `/mnt/sf_myfiles` on Solaris.

  The guest property `/VirtualBox/GuestAdd/SharedFolders/MountPrefix` determines the prefix that is used. Change that guest property to a value other than "sf" to change that prefix; see chapter 4.7, *Guest properties*, page 74 for details.

  > **Note:** Access to auto-mounted shared folders is only granted to the user group `vboxsf`, which is created by the VirtualBox Guest Additions installer. Hence guest users have to be member of that group to have read/write access or to have read-only access in case the folder is not mapped writable.

  To change the mount directory to something other than `/media`, you can set the guest property `/VirtualBox/GuestAdd/SharedFolders/MountDir`.

- **Solaris guests** behave like Linux guests except that `/mnt` is used as the default mount directory instead of `/media`.

To have any changes to auto-mounted shared folders applied while a VM is running, the guest OS needs to be rebooted. (This applies only to auto-mounted shared folders, not the ones which are mounted manually.)

4.4 Drag and Drop

Starting with version 5.0, VirtualBox supports to drag and drop content from the host to the guest and vice versa. For this to work the latest Guest Additions must be installed on the guest.

Drag and drop transparently allows copying or opening files, directories and even certain clipboard formats from one end to the other, e.g. from the host to the guest or from the guest to the host. One then can perform drag and drop operations between the host and a VM as it would be a native drag and drop operation on the host OS.

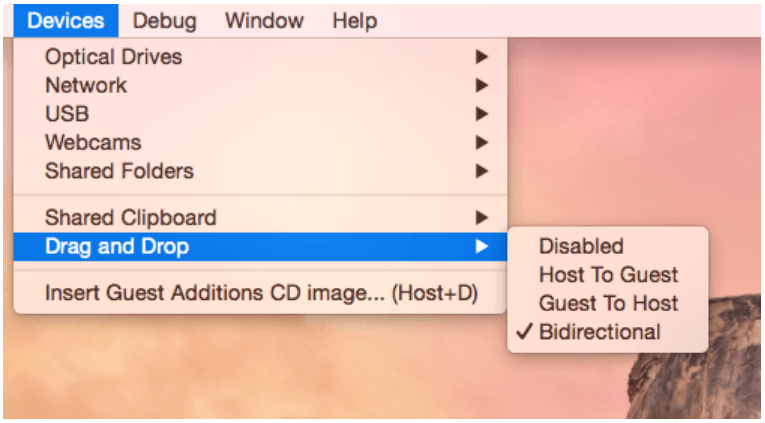
At the moment drag and drop is implemented for Windows- and X-Windows-based systems, both, on host and guest side. As X-Windows sports different drag and drop protocols only the most used one, XDND, is supported for now. Applications using other protocols (such as Motif or OffiX) will not be recognized by VirtualBox.

In context of using drag and drop the origin of the data is called **source**, that is, where the actual data comes from and is specified. On the other hand there is the **target**, which specifies where the data from the source should go to. Transferring data from the source to the target can be done in various ways, e.g. copying, moving or linking.[1]

When transferring data from the host to the guest OS, the host in this case is the source, whereas the guest OS is the target. However, when doing it the other way around, that is, transferring data from the guest OS to the host, the guest OS this time became the source and the host is the target.

For security reasons drag and drop can be configured at runtime on a per-VM basis either using the "Drag and Drop" menu item in the "Devices" menu of the virtual machine or VBoxManage: The following four modes are available:

[1] At the moment only copying of data is supported. Moving or linking is not yet implemented.



- **Disabled** disables the drag and drop entirely. This is the default when creating new VMs.
- **Host To Guest** enables performing drag and drop operations from the host to the guest only.
- **Guest To Host** enables performing drag and drop operations from the guest to the host only.
- **Bidirectional** enables performing drag and drop operations to both directions, e.g. from the host to the guest and vice versa.

> **Note:** Drag and drop support depends on the frontend being used; at the moment only the VirtualBox Manager frontend provides this functionality.

To use VBoxManage for controlling the current drag and drop mode, see chapter 8, *VBoxManage*, page 114. The commands `modifyvm` and `controlvm` allow setting the VM's current drag and drop mode via command line.

4.4.1 Supported formats

As VirtualBox can run on a variety of host OSes and also supports a wide range of guests, certain data formats must be translated after those got transferred over so that the target OS (that is, the side which receiving the data) is able to handle them in an appropriate manner.

> **Note:** When dragging files however, no data conversion is done in any way, e.g. when transferring a file from a Linux guest to a Windows host the Linux-specific line endings won't be converted to Windows ones.

The following formats are handled by the VirtualBox drag and drop service:

- **Plain text**, from applications such as text editors, internet browsers and terminal windows
- **Files**, from file managers such as Windows explorer, Nautilus and Finder
- **Directories**, where the same applies as for files

4.4.2 Known limitations

The following limitations are known:

- On Windows hosts, dragging and dropping content from **UAC-elevated (User Account Control)** programs to non-UAC-elevated programs and vice versa is now allowed. So when starting VirtualBox with Administrator privileges then drag and drop will not work with the Windows Explorer which runs with regular user privileges by default.

4.5 Hardware-accelerated graphics

4.5.1 Hardware 3D acceleration (OpenGL and Direct3D 8/9)

The VirtualBox Guest Additions contain experimental hardware 3D support for Windows, Linux and Solaris guests.[2]

With this feature, if an application inside your virtual machine uses 3D features through the OpenGL or Direct3D 8/9 programming interfaces, instead of emulating them in software (which would be slow), VirtualBox will attempt to use your host's 3D hardware. This works for all supported host platforms (Windows, Mac, Linux, Solaris), provided that your host operating system can make use of your accelerated 3D hardware in the first place.

The 3D acceleration currently has the following preconditions:

1. It is only available for certain Windows, Linux and Solaris guests. In particular:
   - 3D acceleration with Windows guests requires Windows 2000, Windows XP, Vista or Windows 7. Both OpenGL and Direct3D 8/9 (not with Windows 2000) are supported (experimental).
   - OpenGL on Linux requires kernel 2.6.27 and higher as well as X.org server version 1.5 and higher. Ubuntu 10.10 and Fedora 14 have been tested and confirmed as working.
   - OpenGL on Solaris guests requires X.org server version 1.5 and higher.
2. The Guest Additions must be installed.

   Note: For the basic Direct3D acceleration to work in a Windows Guest, VirtualBox needs to replace Windows system files in the virtual machine. As a result, the Guest Additions installation program offers Direct3D acceleration as an option that must be explicitly enabled. Also, you must install the Guest Additions in "Safe Mode". This does **not** apply to the experimental WDDM Direct3D video driver available for Vista and Windows 7 guests, see chapter 14, *Known limitations*, page 269 for details.

3. Because 3D support is still experimental at this time, it is disabled by default and must be **manually enabled** in the VM settings (see chapter 3.3, *General settings*, page 48).

   Note: Untrusted guest systems should not be allowed to use VirtualBox's 3D acceleration features, just as untrusted host software should not be allowed to use 3D acceleration. Drivers for 3D hardware are generally too complex to be made properly secure and any software which is allowed to access them may be able to compromise the operating system running them. In addition, enabling 3D acceleration gives the guest direct access to a large body of additional program code in the VirtualBox host process which it might conceivably be able to use to crash the virtual machine.

[2] OpenGL support for Windows guests was added with VirtualBox 2.1; support for Linux and Solaris followed with VirtualBox 2.2. With VirtualBox 3.0, Direct3D 8/9 support was added for Windows guests. OpenGL 2.0 is now supported as well. With VirtualBox 4.1 Windows Aero theme support is added for Windows Vista and Windows 7 guests (experimental)

With VirtualBox 4.1, Windows Aero theme support is added for Windows Vista and Windows 7 guests. To enable Aero theme support, the experimental VirtualBox WDDM video driver must be installed, which is available with the Guest Additions installation. Since the WDDM video driver is still experimental at this time, it is not installed by default and must be **manually selected** in the Guest Additions installer by answering "No" int the "Would you like to install basic Direct3D support" dialog displayed when the Direct3D feature is selected.

> **Note:** Unlike the current basic Direct3D support, the WDDM video driver installation does **not** require the "Safe Mode".

The Aero theme is not enabled by default. To enable it

- In Windows Vista guest: right-click on the desktop, in the context menu select "Personalize", then select "Windows Color and Appearance" in the "Personalization" window, in the "Appearance Settings" dialog select "Windows Aero" and press "OK"
- In Windows 7 guest: right-click on the desktop, in the context menu select "Personalize" and select any Aero theme in the "Personalization" window

Technically, VirtualBox implements this by installing an additional hardware 3D driver inside your guest when the Guest Additions are installed. This driver acts as a hardware 3D driver and reports to the guest operating system that the (virtual) hardware is capable of 3D hardware acceleration. When an application in the guest then requests hardware acceleration through the OpenGL or Direct3D programming interfaces, these are sent to the host through a special communication tunnel implemented by VirtualBox, and then the *host* performs the requested 3D operation via the host's programming interfaces.

4.5.2 Hardware 2D video acceleration for Windows guests

Starting with version 3.1, the VirtualBox Guest Additions contain experimental hardware 2D video acceleration support for Windows guests.

With this feature, if an application (e.g. a video player) inside your Windows VM uses 2D video overlays to play a movie clip, then VirtualBox will attempt to use your host's video acceleration hardware instead of performing overlay stretching and color conversion in software (which would be slow). This currently works for Windows, Linux and Mac host platforms, provided that your host operating system can make use of 2D video acceleration in the first place.

The 2D video acceleration currently has the following preconditions:

1. It is only available for Windows guests (XP or later).
2. The Guest Additions must be installed.
3. Because 2D support is still experimental at this time, it is disabled by default and must be **manually enabled** in the VM settings (see chapter 3.3, *General settings*, page 48).

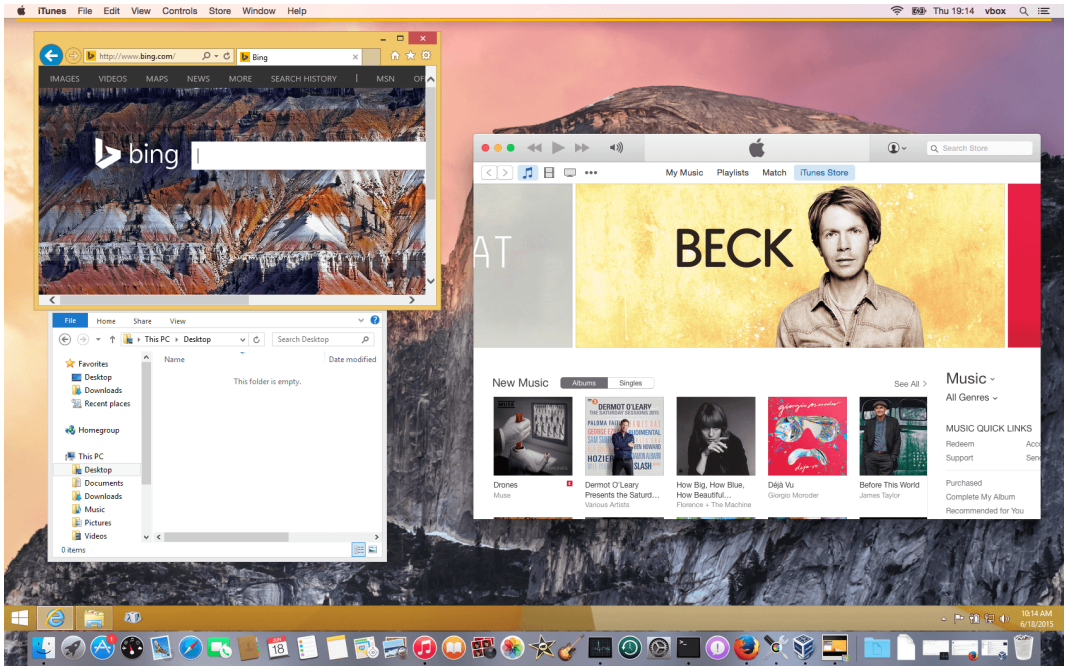
Technically, VirtualBox implements this by exposing video overlay DirectDraw capabilities in the Guest Additions video driver. The driver sends all overlay commands to the host through a special communication tunnel implemented by VirtualBox. On the host side, OpenGL is then used to implement color space transformation and scaling

4.6 Seamless windows

With the "seamless windows" feature of 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);
- Supported Linux or Solaris guests running the X Window System (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.

4.7 Guest properties

Starting with version 2.1, VirtualBox allows for requesting certain properties from a running guest, provided that the VirtualBox Guest Additions are installed and the VM is running. This is good for two things:

1. A number of predefined VM characteristics are automatically maintained by VirtualBox and can be retrieved on the host, e.g. to monitor VM performance and statistics.
2. In addition, arbitrary string data can be exchanged between guest and host. This works in both directions.

To accomplish this, VirtualBox establishes a private communication channel between the VirtualBox Guest Additions and the host, and software on both sides can use this channel to exchange string data for arbitrary purposes. Guest properties are simply string keys to which a value is attached. They can be set (written to) by either the host and the guest, and they can also be read from both sides.

In addition to establishing the general mechanism of reading and writing values, a set of predefined guest properties is automatically maintained by the VirtualBox Guest Additions to

allow for retrieving interesting guest data such as the guest's exact operating system and service pack level, the installed version of the Guest Additions, users that are currently logged into the guest OS, network statistics and more. These predefined properties are all prefixed with `/VirtualBox/` and organized into a hierarchical tree of keys.

Some of this runtime information is shown when you select "Session Information Dialog" from a virtual machine's "Machine" menu.

A more flexible way to use this channel is via the `VBoxManage guestproperty` command set; see chapter 8.33, *VBoxManage guestproperty*, page 164 for details. For example, to have *all* the available guest properties for a given running VM listed with their respective values, use this:

```
$ VBoxManage guestproperty enumerate "Windows Vista III"
VirtualBox Command Line Management Interface Version 5.1.22
(C) 2005-2017 Oracle Corporation
All rights reserved.

Name: /VirtualBox/GuestInfo/OS/Product, value: Windows Vista Business Edition,
    timestamp: 1229098278843087000, flags:
Name: /VirtualBox/GuestInfo/OS/Release, value: 6.0.6001,
    timestamp: 1229098278950553000, flags:
Name: /VirtualBox/GuestInfo/OS/ServicePack, value: 1,
    timestamp: 1229098279122627000, flags:
Name: /VirtualBox/GuestAdd/InstallDir,
    value: C:/Program Files/Oracle/VirtualBox
    Guest Additions, timestamp: 1229098279269739000, flags:
Name: /VirtualBox/GuestAdd/Revision, value: 40720,
    timestamp: 1229098279345664000, flags:
Name: /VirtualBox/GuestAdd/Version, value: 5.1.22,
    timestamp: 1229098279479515000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxControl.exe, value: 5.1.22r40720,
    timestamp: 1229098279651731000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxHook.dll, value: 5.1.22r40720,
    timestamp: 1229098279804835000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxDisp.dll, value: 5.1.22r40720,
    timestamp: 1229098279880611000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxMRXNP.dll, value: 5.1.22r40720,
    timestamp: 1229098279882618000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxService.exe, value: 5.1.22r40720,
    timestamp: 1229098279883195000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxTray.exe, value: 5.1.22r40720,
    timestamp: 1229098279885027000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxGuest.sys, value: 5.1.22r40720,
    timestamp: 1229098279886838000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxMouse.sys, value: 5.1.22r40720,
    timestamp: 1229098279890600000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxSF.sys, value: 5.1.22r40720,
    timestamp: 1229098279893056000, flags:
Name: /VirtualBox/GuestAdd/Components/VBoxVideo.sys, value: 5.1.22r40720,
    timestamp: 1229098279895767000, flags:
Name: /VirtualBox/GuestInfo/OS/LoggedInUsers, value: 1,
    timestamp: 1229099826317660000, flags:
Name: /VirtualBox/GuestInfo/OS/NoLoggedInUsers, value: false,
    timestamp: 1229098455580553000, flags:
Name: /VirtualBox/GuestInfo/Net/Count, value: 1,
    timestamp: 1229099826299785000, flags:
Name: /VirtualBox/HostInfo/GUI/LanguageID, value: C,
    timestamp: 1229098151272771000, flags:
Name: /VirtualBox/GuestInfo/Net/0/V4/IP, value: 192.168.2.102,
    timestamp: 1229099826300088000, flags:
Name: /VirtualBox/GuestInfo/Net/0/V4/Broadcast, value: 255.255.255.255,
    timestamp: 1229099826300220000, flags:
Name: /VirtualBox/GuestInfo/Net/0/V4/Netmask, value: 255.255.255.0,
    timestamp: 1229099826300350000, flags:
Name: /VirtualBox/GuestInfo/Net/0/Status, value: Up,
    timestamp: 1229099826300524000, flags:
Name: /VirtualBox/GuestInfo/OS/LoggedInUsersList, value: username,
    timestamp: 1229099826317386000, flags:
```

To query the value of a single property, use the "get" subcommand like this:

```
$ VBoxManage guestproperty get "Windows Vista III" "/VirtualBox/GuestInfo/OS/Product"
VirtualBox Command Line Management Interface Version 5.1.22
(C) 2005-2017 Oracle Corporation
All rights reserved.

Value: Windows Vista Business Edition
```

To add or change guest properties from the guest, use the tool `VBoxControl`. This tool is included in the Guest Additions of VirtualBox 2.2 or later. When started from a Linux guest, this tool requires root privileges for security reasons:

```
$ sudo VBoxControl guestproperty enumerate
VirtualBox Guest Additions Command Line Management Interface Version 5.1.22
(C) 2009-2017 Oracle Corporation
All rights reserved.

Name: /VirtualBox/GuestInfo/OS/Release, value: 2.6.28-18-generic,
    timestamp: 1265813265835667000, flags: <NULL>
Name: /VirtualBox/GuestInfo/OS/Version, value: #59-Ubuntu SMP Thu Jan 28 01:23:03 UTC 2010,
    timestamp: 1265813265836305000, flags: <NULL>
      ...
```

For more complex needs, you can use the VirtualBox programming interfaces; see chapter 11, *VirtualBox programming interfaces*, page 245.

4.8 Guest control

Starting with version 3.2, the Guest Additions of VirtualBox allow starting applications inside a VM from the host system.

For this to work, the application needs to be installed inside the guest; no additional software needs to be installed on the host. Additionally, text mode output (to stdout and stderr) can be shown on the host for further processing along with options to specify user credentials and a timeout value (in milliseconds) to limit time the application is able to run.

This feature can be used to automate deployment of software within the guest.

Starting with version 4.0, the Guest Additions for Windows allow for automatic updating (only already installed Guest Additions 4.0 or later). Also, copying files from host to the guest as well as remotely creating guest directories is available.

To use these features, use the VirtualBox command line, see chapter 8.34, *VBoxManage guestcontrol*, page 165.

4.9 Memory overcommitment

In server environments with many VMs; the Guest Additions can be used to share physical host memory between several VMs, reducing the total amount of memory in use by the VMs. If memory usage is the limiting factor and CPU resources are still available, this can help with packing more VMs on each host.

4.9.1 Memory ballooning

Starting with version 3.2, the Guest Additions of VirtualBox can change the amount of host memory that a VM uses while the machine is running. Because of how this is implemented, this feature is called "memory ballooning".

Note: VirtualBox supports memory ballooning only on 64-bit hosts, and it is not supported on Mac OS X hosts.

Note: Memory ballooning does not work with large pages enabled. To turn off large pages support for a VM, run `VBoxManage modifyvm <VM name> --largepages off`

Normally, to change the amount of memory allocated to a virtual machine, one has to shut down the virtual machine entirely and modify its settings. With memory ballooning, memory that was allocated for a virtual machine can be given to another virtual machine without having to shut the machine down.

When memory ballooning is requested, the VirtualBox Guest Additions (which run inside the guest) allocate physical memory from the guest operating system on the kernel level and lock this memory down in the guest. This ensures that the guest will not use that memory any longer: no guest applications can allocate it, and the guest kernel will not use it either. VirtualBox can then re-use this memory and give it to another virtual machine.

The memory made available through the ballooning mechanism is only available for re-use by VirtualBox. It is *not* returned as free memory to the host. Requesting balloon memory from a running guest will therefore not increase the amount of free, unallocated memory on the host. Effectively, memory ballooning is therefore a memory overcommitment mechanism for multiple virtual machines while they are running. This can be useful to temporarily start another machine, or in more complicated environments, for sophisticated memory management of many virtual machines that may be running in parallel depending on how memory is used by the guests.

At this time, memory ballooning is only supported through VBoxManage. Use the following command to increase or decrease the size of the memory balloon within a running virtual machine that has Guest Additions installed:

```
VBoxManage controlvm "VM name" guestmemoryballoon <n>
```

where `"VM name"` is the name or UUID of the virtual machine in question and `<n>` is the amount of memory to allocate from the guest in megabytes. See chapter 8.13, *VBoxManage controlvm*, page 143 for more information.

You can also set a default balloon that will automatically be requested from the VM every time after it has started up with the following command:

```
VBoxManage modifyvm "VM name" --guestmemoryballoon <n>
```

By default, no balloon memory is allocated. This is a VM setting, like other `modifyvm` settings, and therefore can only be set while the machine is shut down; see chapter 8.8, *VBoxManage modifyvm*, page 128.

4.9.2 Page Fusion

Whereas memory ballooning simply reduces the amount of RAM that is available to a VM, Page Fusion works differently: it avoids memory duplication between several similar running VMs.

In a server environment running several similar VMs (e.g. with identical operating systems) on the same host, lots of memory pages are identical. VirtualBox's Page Fusion technology, introduced with VirtualBox 3.2, is a novel technique to efficiently identify these identical memory pages and share them between multiple VMs.

Note: VirtualBox supports Page Fusion only on 64-bit hosts, and it is not supported on Mac OS X hosts. Page Fusion currently works only with Windows guests (2000 and later).

The more similar the VMs on a given host are, the more efficiently Page Fusion can reduce the amount of host memory that is in use. It therefore works best if all VMs on a host run identical operating systems (e.g. Windows XP Service Pack 2). Instead of having a complete copy of each operating system in each VM, Page Fusion identifies the identical memory pages in use by these operating systems and eliminates the duplicates, sharing host memory between several machines ("deduplication"). If a VM tries to modify a page that has been shared with other VMs, a new page is allocated again for that VM with a copy of the shared page ("copy on write"). All this is fully transparent to the virtual machine.

You may be familiar with this kind of memory overcommitment from other hypervisor products, which call this feature "page sharing" or "same page merging". However, Page Fusion differs significantly from those other solutions, whose approaches have several drawbacks:

1. Traditional hypervisors scan *all* guest memory and compute checksums (hashes) for every single memory page. Then, they look for pages with identical hashes and compare the entire content of those pages; if two pages produce the same hash, it is very likely that the pages are identical in content. This, of course, can take rather long, especially if the system is not idling. As a result, the additional memory only becomes available after a significant amount of time (this can be hours or even days!). Even worse, this kind of page sharing algorithm generally consumes significant CPU resources and increases the virtualization overhead by 10-20%.

   Page Fusion in VirtualBox uses logic in the VirtualBox Guest Additions to quickly identify memory cells that are most likely identical across VMs. It can therefore achieve most of the possible savings of page sharing almost immediately and with almost no overhead.

2. Page Fusion is also much less likely to be confused by identical memory that it will eliminate just to learn seconds later that the memory will now change and having to perform a highly expensive and often service-disrupting reallocation.

At this time, Page Fusion can only be controlled with VBoxManage, and only while a VM is shut down. To enable Page Fusion for a VM, use the following command:

```
VBoxManage modifyvm "VM name" --pagefusion on
```

You can observe Page Fusion operation using some metrics. `RAM/VMM/Shared` shows the total amount of fused pages, whereas the per-VM metric `Guest/RAM/Usage/Shared` will return the amount of fused memory for a given VM. Please refer to chapter 8.35, *VBoxManage metrics*, page 172 for information on how to query metrics.

> **Note:** Enabling Page Fusion might indirectly increase the chances for malicious guests to successfully attack other VMs running on the same host, see chapter 13.3.4, *Potentially insecure operations*, page 267.

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, *Disk image files (VDI, VMDK, VHD, HDD)*, page 82.

2. Alternatively, if you have iSCSI storage servers, you can attach such a server to VirtualBox as well; this is described in chapter 5.10, *iSCSI servers*, page 91.

3. Finally, as an advanced feature, you can allow a virtual machine to access one of your host disks directly; this advanced feature is described in chapter 9.9.1, *Using a raw host hard disk from a guest*, page 198.

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), SCSI, SAS, USB MSD, NVMe

In a real PC, hard disks and CD/DVD drives are connected to a device called hard disk controller which drives hard disk operation and data transfers. VirtualBox can emulate the five most common types of hard disk controllers typically found in today's PCs: IDE, SATA (AHCI), SCSI, SAS, USB-based and NVMe mass storage devices.[1]

- **IDE (ATA)** controllers are a backwards compatible yet very advanced extension of the disk controller in the IBM PC/AT (1984). Initially, this interface worked only with hard disks, but was later extended to also support CD-ROM drives and other types of removable media. In physical PCs, this standard uses flat ribbon parallel cables with 40 or 80 wires. Each such cable can connect two devices to a controller, which have traditionally been called "master" and "slave". Typical PCs had two connectors for such cables; as a result, support for up to four IDE devices was most common.

  In VirtualBox, each virtual machine may have one IDE controller enabled, which gives you up to four virtual storage devices that you can attach to the machine. (By default, one of these four – the secondary master – is preconfigured to be the machine's virtual CD/DVD drive, but this can be changed.[2])

  So even if your guest operating system has no support for SCSI or SATA devices, it should always be able to see an IDE controller.

[1] SATA support was added with VirtualBox 1.6; experimental SCSI support was added with 2.1 and fully implemented with 2.2. Generally, storage attachments were made much more flexible with VirtualBox 3.1; see below. Support for the LSI Logic SAS controller was added with VirtualBox 3.2; USB mass storage devices are supported since VirtualBox 5.0; NVMe controller support was added with VirtualBox 5.1.

[2] The assignment of the machine's CD/DVD drive to the secondary master was fixed before VirtualBox 3.1; it is now changeable, and the drive can be at other slots of the IDE controller, and there can be more than one such drive.

You can also select which exact type of IDE controller hardware VirtualBox should present to the virtual machine (PIIX3, PIIX4 or ICH6). This makes no difference in terms of performance, but if you import a virtual machine from another virtualization product, the operating system in that machine may expect a particular controller type and crash if it isn't found.

After you have created a new virtual machine with the "New Virtual Machine" wizard of the graphical user interface, you will typically see one IDE controller in the machine's "Storage" settings where the virtual CD/DVD drive will be attached to one of the four ports of this controller.

- **Serial ATA (SATA)** is a newer standard introduced in 2003. Compared to IDE, it supports both much higher speeds and more devices per controller. Also, with physical hardware, devices can be added and removed while the system is running. The standard interface for SATA controllers is called Advanced Host Controller Interface (**AHCI**).

  Like a real SATA controller, VirtualBox's virtual SATA controller operates faster and also consumes fewer CPU resources than the virtual IDE controller. Also, this allows you to connect up to 30 virtual hard disks to one machine instead of just three, as with the VirtualBox IDE controller (with the DVD drive already attached).

  For this reason, starting with version 3.2 and depending on the selected guest operating system, VirtualBox uses SATA as the default for newly created virtual machines. One virtual SATA controller is created by default, and the default disk that is created with a new VM is attached to this controller.

> **Warning:** The entire SATA controller and the virtual disks attached to it (including those in IDE compatibility mode) will not be seen by operating systems that do not have device support for AHCI. In particular, **there is no support for AHCI in Windows before Windows Vista**, so Windows XP (even SP3) will not see such disks unless you install additional drivers. It is possible to switch from IDE to SATA after installation by installing the SATA drivers and changing the controller type in the VM settings dialog.[a]
>
> [a] VirtualBox recommends the Intel Matrix Storage drivers which can be downloaded from http://downloadcenter.intel.com/Product_Filter.aspx?ProductID=2101.

  To add a SATA controller to a machine for which it has not been enabled by default (either because it was created by an earlier version of VirtualBox, or because SATA is not supported by default by the selected guest operating system), go to the "Storage" page of the machine's settings dialog, click on the "Add Controller" button under the "Storage Tree" box and then select "Add SATA Controller". After this, the additional controller will appear as a separate PCI device in the virtual machine, and you can add virtual disks to it.

  To change the IDE compatibility mode settings for the SATA controller, please see chapter 8.19, *VBoxManage storagectl*, page 154.

- **SCSI** is another established industry standard, standing for "Small Computer System Interface". SCSI was standardized as early as 1986 as a generic interface for data transfer between all kinds of devices, including storage devices. Today SCSI is still used for connecting hard disks and tape devices, but it has mostly been displaced in commodity hardware. It is still in common use in high-performance workstations and servers.

  Primarily for compatibility with other virtualization software, VirtualBox optionally supports LSI Logic and BusLogic SCSI controllers, to each of which up to 15 virtual hard disks can be attached.

  To enable a SCSI controller, on the "Storage" page of a virtual machine's settings dialog, click on the "Add Controller" button under the "Storage Tree" box and then select "Add

SCSI Controller". After this, the additional controller will appear as a separate PCI device in the virtual machine.

> **Warning:** As with the other controller types, a SCSI controller will only be seen by operating systems with device support for it. Windows 2003 and later ships with drivers for the LSI Logic controller, while Windows NT 4.0 and Windows 2000 ships with drivers for the BusLogic controller. Windows XP ships with drivers for neither.

- **Serial Attached SCSI (SAS)** is another bus standard which uses the SCSI command set. As opposed to SCSI, however, with physical devices, serial cables are used instead of parallel ones, which simplifies physical device connections. In some ways, therefore, SAS is to SCSI what SATA is to IDE: it allows for more reliable and faster connections.

  To support high-end guests which require SAS controllers, VirtualBox emulates a LSI Logic SAS controller, which can be enabled much the same way as a SCSI controller. At this time, up to eight devices can be connected to the SAS controller.

  > **Warning:** As with SATA, the SAS controller will only be seen by operating systems with device support for it. In particular, **there is no support for SAS in Windows before Windows Vista**, so Windows XP (even SP3) will not see such disks unless you install additional drivers.

- The **USB mass storage device class** is a standard to connect external storage devices like hard disks or flash drives to a host through USB. All major operating systems support these devices for a long time and ship generic drivers making third-party drivers superfluous. In particular legacy operating systems without support for SATA controllers may benefit from USB mass storage devices.

  The virtual USB storage controller offered by VirtualBox works different than the other storage controller types: When storage controllers appear as a single PCI device to the guest with multiple disks attached to it, the USB-based storage controller does not appear as virtual storage controller. Each disk attached to the controller appears as a dedicated USB device to the guest.

  > **Warning:** Booting from drives attached via USB is when EFI is used as the BIOS lacks USB support.

- **Non volatile memory express (NVMe)** is a very recent standard which emerged in 2011 connecting non volatile memory (NVM) directly over PCI express to lift the bandwidth limitation of the previously used SATA protocol for SSDs. Unlike other standards the command set is very simple to achieve maximum throughput and is not compatible with ATA or SCSI. Operating systems need to support NVMe devices to make use of them. For example Windows 8.1 added native NVMe support, for Windows 7 native support was added with an update. [3]

  > **Warning:** Booting from drives attached via NVMe is only supported when EFI is used as the BIOS lacks the appropriate driver.

[3] The NVMe controller is part of the extension pack.

In summary, VirtualBox gives you the following categories of virtual storage slots:

1. four slots attached to the traditional IDE controller, which are always present (one of which typically is a virtual CD/DVD drive);
2. 30 slots attached to the SATA controller, if enabled and supported by the guest operating system;
3. 15 slots attached to the SCSI controller, if enabled and supported by the guest operating system;
4. eight slots attached to the SAS controller, if enabled and supported by the guest operating system.
5. eight slots attached to the virtual USB controller, if enabled and supported by the guest operating system.
6. up to 255 slots attached to the NVMe controller, if enabled and supported by the guest operating system.

Given this large choice of storage controllers, you may ask yourself which one to choose. In general, you should avoid IDE unless it is the only controller supported by your guest. Whether you use SATA, SCSI or SAS does not make any real difference. The variety of controllers is only supplied for VirtualBox for compatibility with existing hardware and other hypervisors.

5.2 Disk image files (VDI, VMDK, VHD, HDD)

Disk image files reside on the host system and are seen by the guest systems as hard disks of a certain geometry. When a guest operating system reads from or writes to a hard disk, VirtualBox redirects the request to the image file.

Like a physical disk, a virtual disk has a size (capacity), which must be specified when the image file is created. As opposed to a physical disk however, VirtualBox allows you to expand an image file after creation, even if it has data already; see chapter 8.23, *VBoxManage modifymedium*, page 156 for details.[4]

VirtualBox supports four variants of disk image files:

- Normally, VirtualBox uses its own container format for guest hard disks – Virtual Disk Image (VDI) files. In particular, this format will be used when you create a new virtual machine with a new disk.
- VirtualBox also fully supports the popular and open VMDK container format that is used by many other virtualization products, in particular, by VMware.[5]
- VirtualBox also fully supports the VHD format used by Microsoft.
- Image files of Parallels version 2 (HDD format) are also supported.[6] For lack of documentation of the format, newer formats (3 and 4) are not supported. You can however convert such image files to version 2 format using tools provided by Parallels.

Irrespective of the disk capacity and format, as briefly mentioned in chapter 1.7, *Creating your first virtual machine*, page 18, there are two options of how to create a disk image: fixed-size or dynamically allocated.

[4] Image resizing was added with VirtualBox 4.0.

[5] Initial support for VMDK was added with VirtualBox 1.4; since version 2.1, VirtualBox supports VMDK fully, meaning that you can create snapshots and use all the other advanced features described above for VDI images with VMDK also.

[6] Support was added with VirtualBox 3.1.

- If you create a **fixed-size image**, an image file will be created on your host system which has roughly the same size as the virtual disk's capacity. So, for a 10G disk, you will have a 10G file. Note that the creation of a fixed-size image can take a long time depending on the size of the image and the write performance of your hard disk.

- For more flexible storage management, use a **dynamically allocated image**. This will initially be very small and not occupy any space for unused virtual disk sectors, but will grow every time a disk sector is written to for the first time, until the drive reaches the maximum capacity chosen when the drive was created. While this format takes less space initially, the fact that VirtualBox needs to expand the image file consumes additional computing resources, so until the disk file size has stabilized, write operations may be slower than with fixed size disks. However, after a time the rate of growth will slow and the average penalty for write operations will be negligible.

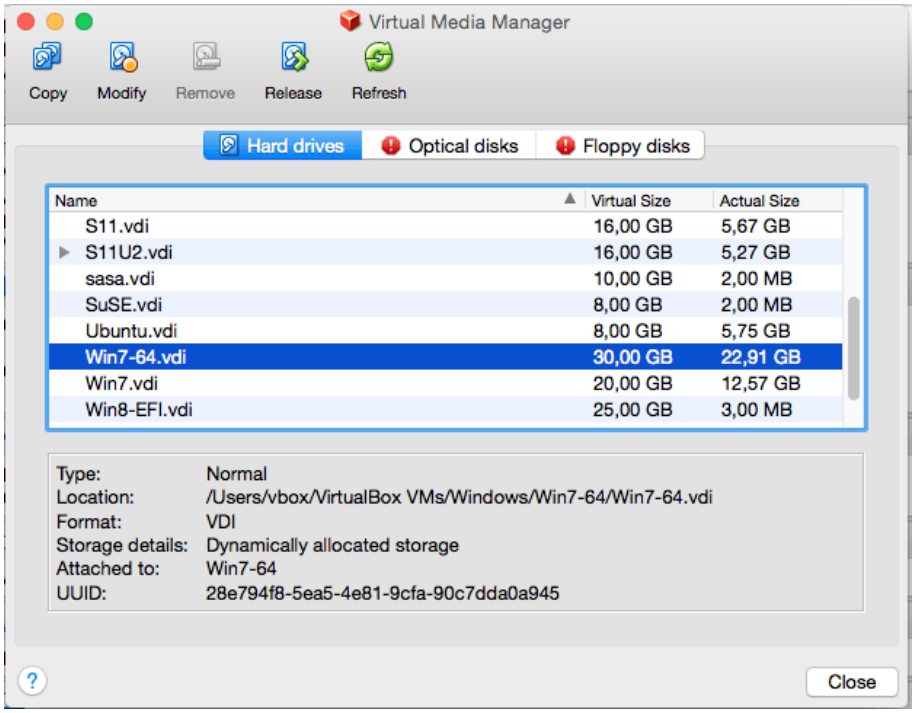
5.3 The Virtual Media Manager

VirtualBox keeps track of all the hard disk, CD/DVD-ROM and floppy disk images which are in use by virtual machines. These are often referred to as "known media" and come from two sources:

- all media currently attached to virtual machines;

- "registered" media for compatibility with VirtualBox versions older than version 4.0. For details about how media registration has changed with version 4.0, please refer to chapter 10.1, *Where VirtualBox stores its files*, page 234.

The known media can be viewed and changed in the **Virtual Media Manager**, which you can access from the "File" menu in the VirtualBox main window:



The known media are 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 third-party formats listed in the previous chapter;
- CD/DVD images in standard ISO format;
- floppy images in standard RAW format.

As you can see in the screenshot above, for each image, the Virtual Media Manager shows you the full path of the image file and other information, such as the virtual machine the image is currently attached to, if any.

The Virtual Media Manager allows you to

- **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.
- **copy** a virtual hard disk, to another one - target type can be different, options are - VDI, VHD or VMDK.
- **modify** the attributes of the disk image file - available options are : Normal, Immutable, Writethrough, Shareable, Multi-attach.
- **refresh** the values for the displayed attributes of the disk image currently selected in the window.

These commands are accessible once a medium has been selected either by selecting from the options shown at the top of the window, or by right-clicking the medium and selecting from the options shown on the drop-down menu.

Starting with version 4.0, to **create new disk images,** please use the "Storage" page in a virtual machine's settings dialog because disk images are now by default stored in each machine's own folder.

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 5.6, *Cloning disk images*, page 88 for instructions on this matter. Also, if you want to copy a virtual machine to another system, VirtualBox has an import/export facility that might be better suited for your needs; see chapter 1.14, *Importing and exporting virtual machines*, page 31.

5.4 Special image write modes

For each virtual disk image supported by VirtualBox, you can determine separately how it should be affected by write operations from a virtual machine and snapshot operations. This applies to all of the aforementioned image formats (VDI, VMDK, VHD or HDD) and irrespective of whether an image is fixed-size or dynamically allocated.

By default, images are in "normal" mode. To mark an existing image with one of the non-standard modes listed below, use `VBoxManage modifyhd`; see chapter 8.23, *VBoxManage modifymedium*, page 156. Alternatively, use VBoxManage to attach the image to a VM and use the `--mtype` argument; see chapter 8.18, *VBoxManage storageattach*, page 150.

1. With **normal images** (the default setting), there are no restrictions on how guests can read from and write to the disk.

   When you take a snapshot of your virtual machine as described in chapter 1.10, *Snapshots*, page 26, 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.

   (Technically, strictly speaking, the image file itself is not "reset". Instead, when a snapshot is taken, VirtualBox "freezes" the image file and no longer writes to it. For the write operations from the VM, a second, "differencing" image file is created which receives only the changes to the original image; see the next section for details.)

   While you can attach the same "normal" image to more than one virtual machine, only one of these virtual machines attached to the same image file can be executed simultaneously, as otherwise there would be conflicts if several machines write to the same image file.[7]

2. By contrast, **write-through hard disks** are completely unaffected by snapshots: their state is *not* saved when a snapshot is taken, and not restored when a snapshot is restored.

3. **Shareable hard disks** are a variant of write-through hard disks. In principle they behave exactly the same, i.e. their state is *not* saved when a snapshot is taken, and not restored when a snapshot is restored. The difference only shows if you attach such disks to several VMs. Shareable disks may be attached to several VMs which may run concurrently. This makes them suitable for use by cluster filesystems between VMs and similar applications which are explicitly prepared to access a disk concurrently. Only fixed size images can be used in this way, and dynamically allocated images are rejected.

   Warning: This is an expert feature, and misuse can lead to data loss – regular filesystems are not prepared to handle simultaneous changes by several parties.

4. Next, **immutable images** only remember write accesses temporarily while the virtual machine is running; all changes are lost when the virtual machine is powered on the next time. As a result, as opposed to "normal" images, the same immutable image can be used with several virtual machines without restrictions.

   Creating an immutable image makes little sense since it would be initially empty and lose its contents with every machine restart (unless you really want to have a disk that is always unformatted when the machine starts up). As a result, normally, you would first create a "normal" image and then, when you deem its contents useful, later mark it immutable.

   If you take a snapshot of a machine with immutable images, then on every machine power-up, those images are reset to the state of the last (current) snapshot (instead of the state of the original immutable image).

   Note: As a special exception, immutable images are *not* reset if they are attached to a machine in saved state or whose last snapshot was taken while the machine was running (a so-called "online" snapshot). As a result, if the machine's current snapshot is such an "online" snapshot, its immutable images behave exactly like the "normal" images described previously. To re-enable the automatic resetting of such images, delete the current snapshot of the machine.

[7] This restriction is more lenient now than it was before VirtualBox 2.2. Previously, each "normal" disk image could only be *attached* to one single machine. Now it can be attached to more than one machine so long as only one of these machines is running.

Again, technically, VirtualBox never writes to an immutable image directly at all. All write operations from the machine will be directed to a differencing image; the next time the VM is powered on, the differencing image is reset so that every time the VM starts, its immutable images have exactly the same content.[8] The differencing image is only reset when the machine is powered on from within VirtualBox, not when you reboot by requesting a reboot from within the machine. This is also why immutable images behave as described above when snapshots are also present, which use differencing images as well.

If the automatic discarding of the differencing image on VM startup does not fit your needs, you can turn it off using the `autoreset` parameter of `VBoxManage modifyhd`; see chapter 8.23, *VBoxManage modifymedium*, page 156 for details.

5. An image in **multiattach mode** can be attached to more than one virtual machine at the same time, even if these machines are running simultaneously. For each virtual machine to which such an image is attached, a differencing image is created. As a result, data that is written to such a virtual disk by one machine is not seen by the other machines to which the image is attached; each machine creates its own write history of the multiattach image.

   Technically, a "multiattach" image behaves identically to an "immutable" image except the differencing image is not reset every time the machine starts.

   This mode is useful for sharing files which are almost never written, for instance picture galleries, where every guest changes only a small amount of data and the majority of the disk content remains unchanged. The modified blocks are stored in differencing images which remain relatively small and the shared content is stored only once at the host.

6. Finally, the **read-only image** is used automatically for CD/DVD images, since CDs/DVDs can never be written to.

To illustrate the differences between the various types with respect to snapshots: Assume 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 restore the snapshot, and the earlier state of your hard disk image will be restored as well (and your virus infection will be undone). With an immutable hard disk, all it takes is to shut down and power on your VM, and the virus infection will be discarded. With a write-through image however, you cannot easily undo the virus infection by means of virtualization, but will have to disinfect your virtual machine like a real computer.

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

5.5 Differencing images

The previous section hinted at differencing images and how they are used with snapshots, immutable images and multiple disk attachments. For the inquisitive VirtualBox user, this section describes in more detail how they work.

A differencing image is a special disk image that only holds the differences to another image. A differencing image by itself is useless, it must always refer to another image. The differencing image is then typically referred to as a "child", which holds the differences to its "parent".

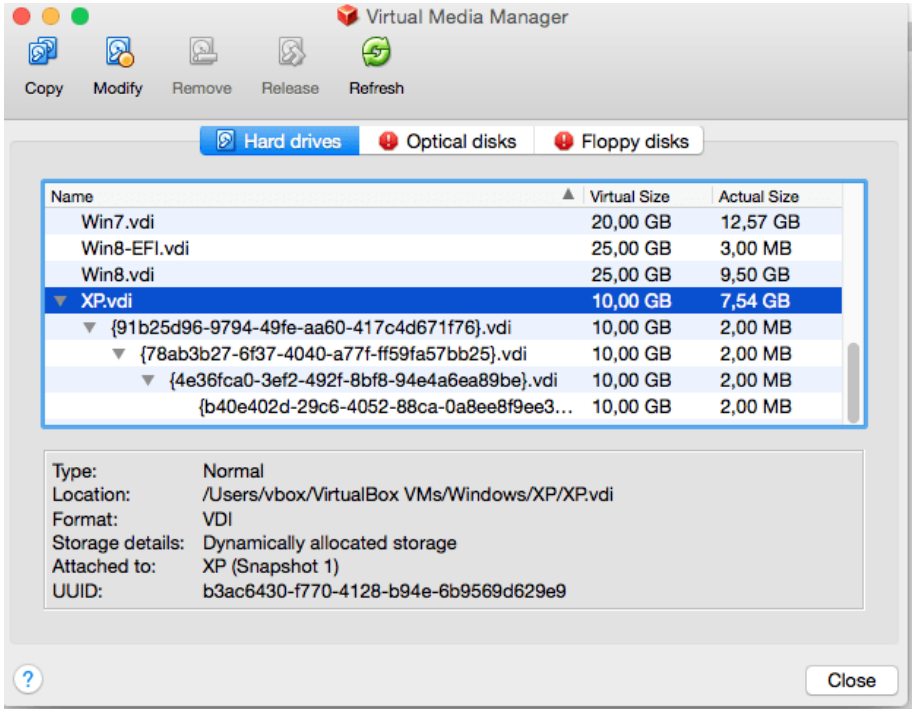
When a differencing image is active, it receives all write operations from the virtual machine instead of its parent. The differencing image only contains the sectors of the virtual hard disk that have changed since the differencing image was created. When the machine reads a sector

[8] This behavior also changed with VirtualBox 2.2. Previously, the differencing images were discarded when the machine session *ended*; now they are discarded every time the machine is powered on.

from such a virtual hard disk, it looks into the differencing image first. If the sector is present, it is returned from there; if not, VirtualBox looks into the parent. In other words, the parent becomes "read-only"; it is never written to again, but it is read from if a sector has not changed.

Differencing images can be chained. If another differencing image is created for a virtual disk that already has a differencing image, then it becomes a "grandchild" of the original parent. The first differencing image then becomes read-only as well, and write operations only go to the second-level differencing image. When reading from the virtual disk, VirtualBox needs to look into the second differencing image first, then into the first if the sector was not found, and then into the original image.

There can be an unlimited number of differencing images, and each image can have more than one child. As a result, the differencing images can form a complex tree with parents, "siblings" and children, depending on how complex your machine configuration is. Write operations always go to the one "active" differencing image that is attached to the machine, and for read operations, VirtualBox may need to look up all the parents in the chain until the sector in question is found. You can look at such a tree in the Virtual Media Manager:



In all of these situations, from the point of view of the virtual machine, the virtual hard disk behaves like any other disk. While the virtual machine is running, there is a slight run-time I/O overhead because VirtualBox might need to look up sectors several times. This is not noticeable however since the tables with sector information are always kept in memory and can be looked up quickly.

Differencing images are used in the following situations:

1. **Snapshots.** When you create a snapshot, as explained in the previous section, VirtualBox "freezes" the images attached to the virtual machine and creates differencing images for each of them (to be precise: one for each image that is not in "write-through" mode). From the point of view of the virtual machine, the virtual disks continue to operate before, but all write operations go into the differencing images. Each time you create another snapshot, for each hard disk attachment, another differencing image is created and attached, forming a chain or tree.

   In the above screenshot, you see that the original disk image is now attached to a snapshot, representing the state of the disk when the snapshot was taken.

If you now **restore** a snapshot – that is, if you want to go back to the exact machine state that was stored in the snapshot –, the following happens:

a) VirtualBox copies the virtual machine settings that were copied into the snapshot back to the virtual machine. As a result, if you have made changes to the machine configuration since taking the snapshot, they are undone.

b) If the snapshot was taken while the machine was running, it contains a saved machine state, and that state is restored as well; after restoring the snapshot, the machine will then be in "Saved" state and resume execution from there when it is next started. Otherwise the machine will be in "Powered Off" state and do a full boot.

c) For each disk image attached to the machine, the differencing image holding all the write operations since the current snapshot was taken is thrown away, and the original parent image is made active again. (If you restored the "root" snapshot, then this will be the root disk image for each attachment; otherwise, some other differencing image descended from it.) This effectively restores the old machine state.

If you later **delete** a snapshot in order to free disk space, for each disk attachment, one of the differencing images becomes obsolete. In this case, the differencing image of the disk attachment cannot simply be deleted. Instead, VirtualBox needs to look at each sector of the differencing image and needs to copy it back into its parent; this is called "merging" images and can be a potentially lengthy process, depending on how large the differencing image is. It can also temporarily need a considerable amount of extra disk space, before the differencing image obsoleted by the merge operation is deleted.

2. **Immutable images.** When an image is switched to "immutable" mode, a differencing image is created as well. As with snapshots, the parent image then becomes read-only, and the differencing image receives all the write operations. Every time the virtual machine is started, all the immutable images which are attached to it have their respective differencing image thrown away, effectively resetting the virtual machine's virtual disk with every restart.

5.6 Cloning disk images

You can duplicate hard disk image files on the same host to quickly produce a second virtual machine with the same operating system setup. However, you should *only* make copies of virtual disk images using the utility supplied with VirtualBox; see chapter 8.24, *VBoxManage clonemedium*, page 157. 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 re-import 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 this:

```
scsi-SATA_VBOX_HARDDISK_VB5cfdb1e2-c251e503
```

The ID for the copied image can be determined with

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

5.7 Host I/O caching

Starting with version 3.2, VirtualBox can optionally disable the I/O caching that the host operating system would otherwise perform on disk image files.

Traditionally, VirtualBox has opened disk image files as normal files, which results in them being cached by the host operating system like any other file. The main advantage of this is speed: when the guest OS writes to disk and the host OS cache uses delayed writing, the write operation can be reported as completed to the guest OS quickly while the host OS can perform the operation asynchronously. Also, when you start a VM a second time and have enough memory available for the OS to use for caching, large parts of the virtual disk may be in system memory, and the VM can access the data much faster.

Note that this applies only to image files; buffering never occurred for virtual disks residing on remote iSCSI storage, which is the more common scenario in enterprise-class setups (see chapter 5.10, *iSCSI servers*, page 91).

While buffering is a useful default setting for virtualizing a few machines on a desktop computer, there are some disadvantages to this approach:

1. Delayed writing through the host OS cache is less secure. When the guest OS writes data, it considers the data written even though it has not yet arrived on a physical disk. If for some reason the write does not happen (power failure, host crash), the likelihood of data loss increases.

2. Disk image files tend to be very large. Caching them can therefore quickly use up the entire host OS cache. Depending on the efficiency of the host OS caching, this may slow down the host immensely, especially if several VMs run at the same time. For example, on Linux hosts, host caching may result in Linux delaying all writes until the host cache is nearly full and then writing out all these changes at once, possibly stalling VM execution for minutes. This can result in I/O errors in the guest as I/O requests time out there.

3. Physical memory is often wasted as guest operating systems typically have their own I/O caches, which may result in the data being cached twice (in both the guest and the host caches) for little effect.

If you decide to disable host I/O caching for the above reasons, VirtualBox uses its own small cache to buffer writes, but no read caching since this is typically already performed by the guest OS. In addition, VirtualBox fully supports asynchronous I/O for its virtual SATA, SCSI and SAS controllers through multiple I/O threads.

Since asynchronous I/O is not supported by IDE controllers, for performance reasons, you may want to leave host caching enabled for your VM's virtual IDE controllers.

For this reason, VirtualBox allows you to configure whether the host I/O cache is used for each I/O controller separately. Either uncheck the "Use host I/O cache" box in the "Storage" settings for a given virtual storage controller, or use the following VBoxManage command to disable the host I/O cache for a virtual storage controller:

```
VBoxManage storagectl "VM name" --name <controllername> --hostiocache off
```

See chapter 8.19, *VBoxManage storagectl*, page 154 for details.

For the above reasons also, VirtualBox now uses SATA controllers by default for new virtual machines.

5.8 Limiting bandwidth for disk images

Starting with version 4.0, VirtualBox allows for limiting the maximum bandwidth used for asynchronous I/O. Additionally it supports sharing limits through bandwidth groups for several images. It is possible to have more than one such limit.

Limits are configured through `VBoxManage`. The example below creates a bandwidth group named "Limit", sets the limit to 20 MB/s and assigns the group to the attached disks of the VM:

```
VBoxManage bandwidthctl "VM name" add Limit --type disk --limit 20M
VBoxManage storageattach "VM name" --storagectl "SATA" --port 0 --device 0 --type hdd
                                   --medium disk1.vdi --bandwidthgroup Limit
VBoxManage storageattach "VM name" --storagectl "SATA" --port 1 --device 0 --type hdd
                                   --medium disk2.vdi --bandwidthgroup Limit
```

All disks in a group share the bandwidth limit, meaning that in the example above the bandwidth of both images combined can never exceed 20 MB/s. However, if one disk doesn't require bandwidth the other can use the remaining bandwidth of its group.

The limits for each group can be changed while the VM is running, with changes being picked up immediately. The example below changes the limit for the group created in the example above to 10 MB/s:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 10M
```

5.9 CD/DVD support

The virtual CD/DVD drive(s) by default support only reading. The medium configuration is changeable at runtime. You can select between three options to provide the medium data:

- **Host Drive** defines that the guest can read from the medium in the host drive.
- **Image file** (typically an ISO file) gives the guest read-only access to the data in the image.
- **Empty** stands for a drive without an inserted medium.

Changing between the above, or changing a medium in the host drive that is accessed by a machine, or changing an image file will signal a medium change to the guest operating system, which can then react to the change (e.g. by starting an installation program).

Medium changes can be prevented by the guest, and VirtualBox reflects that by locking the host drive if appropriate. You can force a medium removal in such situations via the VirtualBox GUI or the VBoxManage command line tool. Effectively this is the equivalent of the emergency eject which many CD/DVD drives provide, with all associated side effects: the guest OS can issue error messages, just like on real hardware, and guest applications may misbehave. Use this with caution.

> **Note:** The identification string of the drive provided to the guest (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.

The standard CD/DVD emulation allows for reading standard data CD and DVD formats only. As an experimental feature, for additional capabilities, it is possible to give the guest direct access to the CD/DVD host drive by enabling "passthrough" mode. Depending on the host hardware, this may enable three things to work, potentially:

- CD/DVD writing from within the guest, if the host DVD drive is a CD/DVD writer;
- playing audio CDs;
- playing encrypted DVDs.

There is a "Passthrough" checkbox in the GUI dialog for configuring the media attached to a storage controller, or you can use the `--passthrough` option with `VBoxManage storageattach`; see chapter 8.18, *VBoxManage storageattach*, page 150 for details.

Even if pass-through is enabled, unsafe commands, such as updating the drive firmware, will be blocked. Video CD formats are never supported, not even in passthrough mode, and cannot be played from a virtual machine.

On Solaris hosts, pass-through requires running VirtualBox with real root permissions due to security measures enforced by the host.

5.10 iSCSI servers

iSCSI stands for "Internet SCSI" and is a standard that allows for using the SCSI protocol over Internet (TCP/IP) connections. Especially with the advent of Gigabit Ethernet, it has become affordable to attach iSCSI storage servers simply as remote hard disks to a computer network. In iSCSI terminology, the server providing storage resources is called an "iSCSI target", while the client connecting to the server and accessing its resources is called "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.

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 use the command line; see chapter 8.18, *VBoxManage storageattach*, page 150.

6 Virtual networking

As briefly mentioned in chapter 3.8, *Network settings*, page 55, VirtualBox provides up to eight 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.

Four of the network cards can be configured in the "Network" section of the settings dialog in the graphical user interface of VirtualBox. You can configure all eight network cards on the command line via VBoxManage modifyvm; see chapter 8.8, *VBoxManage modifyvm*, page 128.

This chapter explains the various networking settings in more detail.

6.1 Virtual networking hardware

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

- AMD PCNet PCI II (Am79C970A);
- AMD PCNet FAST III (Am79C973, the default);
- Intel PRO/1000 MT Desktop (82540EM);
- Intel PRO/1000 T Server (82543GC);
- Intel PRO/1000 MT Server (82545EM);
- Paravirtualized network adapter (virtio-net).

The PCNet FAST III is the default because it is supported by nearly all operating systems out of the box, as well as the GNU GRUB boot manager. As an exception, the Intel PRO/1000 family adapters are chosen for some guest operating system types that no longer ship with drivers for the PCNet card, such as Windows Vista.

The Intel PRO/1000 MT Desktop type works with Windows Vista and later versions. The T Server variant of the Intel PRO/1000 card is recognized by Windows XP guests without additional driver installation. The MT Server variant facilitates OVF imports from other platforms.

The **"Paravirtualized network adapter (virtio-net)"** is special. If you select this, then VirtualBox does *not* virtualize common networking hardware (that is supported by common guest operating systems out of the box). Instead, VirtualBox then expects a special software interface for virtualized environments to be provided by the guest, thus avoiding the complexity of emulating networking hardware and improving network performance. Starting with version 3.1, VirtualBox provides support for the industry-standard "virtio" networking drivers, which are part of the open-source KVM project.

The "virtio" networking drivers are available for the following guest operating systems:

- Linux kernels version 2.6.25 or later can be configured to provide virtio support; some distributions also back-ported virtio to older kernels.

- For Windows 2000, XP and Vista, virtio drivers can be downloaded and installed from the KVM project web page.[1]

VirtualBox also has limited support for so-called **jumbo frames**, i.e. networking packets with more than 1500 bytes of data, provided that you use the Intel card virtualization and bridged networking. In other words, jumbo frames are not supported with the AMD networking devices; in those cases, jumbo packets will silently be dropped for both the transmit and the receive direction. Guest operating systems trying to use this feature will observe this as a packet loss, which may lead to unexpected application behavior in the guest. This does not cause problems with guest operating systems in their default configuration, as jumbo frames need to be explicitly enabled.

6.2 Introduction to networking modes

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

Not attached In this mode, 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.

Network Address Translation (NAT) If all you want is to browse the Web, download files and view e-mail inside the guest, then this default mode should be sufficient for you, and you can safely skip the rest of this section. Please note that there are certain limitations when using Windows file sharing (see chapter 6.3.3, *NAT limitations*, page 95 for details).

NAT Network The NAT network is a new NAT flavour introduced in VirtualBox 4.3. See chapter 6.4, *Network Address Translation Service*, page 96 for details.

Bridged networking This is for more advanced networking needs such as network simulations and running servers in a guest. When enabled, VirtualBox connects to one of your installed network cards and exchanges network packets directly, circumventing your host operating system's network stack.

Internal networking This can be used to create a different kind of software-based network which is visible to selected virtual machines, but not to applications running on the host or to the outside world.

Host-only networking This can be used to create a network containing the host and a set of virtual machines, without the need for the host's physical network interface. Instead, a virtual network interface (similar to a loopback interface) is created on the host, providing connectivity among virtual machines and the host.

Generic networking Rarely used modes share the same generic network interface, by allowing the user to select a driver which can be included with VirtualBox or be distributed in an extension pack.

At the moment there are potentially two available sub-modes:

UDP Tunnel This can be used to interconnect virtual machines running on different hosts directly, easily and transparently, over existing network infrastructure.

VDE (Virtual Distributed Ethernet) networking This option can be used to connect to a Virtual Distributed Ethernet switch on a Linux or a FreeBSD host. At the moment this needs compiling VirtualBox from sources, as the Oracle packages do not include it.

[1] http://www.linux-kvm.org/page/WindowsGuestDrivers.

The following sections describe the available network modes in more detail.

6.3 Network Address Translation (NAT)

Network Address Translation (NAT) is the simplest way of accessing an external network from a virtual machine. Usually, it does not require any configuration on the host network and guest system. For this reason, it is the default networking mode in 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. In VirtualBox this router is placed between each virtual machine and the host. This separation maximizes security since by default virtual machines cannot talk to each other.

The disadvantage of NAT mode is that, much like a private network behind a router, the virtual machine is invisible and unreachable from the outside internet; you cannot run a server this way unless you set up port forwarding (described below).

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.

The virtual machine receives its network address and configuration on the private network from a DHCP server integrated into VirtualBox. The IP address thus assigned to the virtual machine is usually on a completely different network than the host. As more than one card of a virtual machine can be set up to use NAT, the first card is connected to the private network 10.0.2.0, the second card to the network 10.0.3.0 and so on. If you need to change the guest-assigned IP range for some reason, please refer to chapter 9.11, *Fine-tuning the VirtualBox NAT engine*, page 201.

6.3.1 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, like a physical router, VirtualBox can make selected services available to the world outside the guest through **port forwarding.** This means that VirtualBox listens to certain ports on the host and resends all packets which arrive there to the guest, on the same or a different port.

To an application on the host or other physical (or virtual) machines on the network, it looks as though the service being proxied is actually running on the host. This also means that you cannot run the same service on the same ports on the host. However, you still gain the advantages of running the service in a virtual machine – for example, services on the host machine or on other virtual machines cannot be compromised or crashed by a vulnerability or a bug in the service, and the service can run in a different operating system than the host system.

To configure Port Forwarding you can use the graphical Port Forwarding editor which can be found in the Network Settings dialog for Network Adaptors configured to use NAT. Here you can map host ports to guest ports to allow network traffic to be routed to a specific port in the guest.

Alternatively command line tool `VBoxManage` could be used; for details, please refer to chapter 8.8, *VBoxManage modifyvm*, page 128.

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. For example, to set up incoming NAT connections to an `ssh` server in the guest, use the following command:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,,2222,,22"
```

With the above example, all TCP traffic arriving on port 2222 on any host interface will be forwarded to port 22 in the guest. The protocol name `tcp` is a mandatory attribute defining which protocol should be used for forwarding (`udp` could also be used). The name `guestssh` is purely descriptive and will be auto-generated if omitted. The number after `--natpf` denotes the network card, like in other parts of VBoxManage.

To remove this forwarding rule again, use the following command:

```
VBoxManage modifyvm "VM name" --natpf1 delete "guestssh"
```

If for some reason the guest uses a static assigned IP address not leased from the built-in DHCP server, it is required to specify the guest IP when registering the forwarding rule:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,,2222,10.0.2.19,22"
```

This example is identical to the previous one, except that the NAT engine is being told that the guest can be found at the 10.0.2.19 address.

To forward *all* incoming traffic from a specific host interface to the guest, specify the IP of that host interface like this:

```
VBoxManage modifyvm "VM name" --natpf1 "guestssh,tcp,127.0.0.1,2222,,22"
```

This forwards all TCP traffic arriving on the localhost interface (127.0.0.1) via port 2222 to port 22 in the guest.

It is possible to configure incoming NAT connections while the VM is running, see chapter 8.13, *VBoxManage controlvm*, page 143.

6.3.2 PXE booting with NAT

PXE booting is now supported in NAT mode. The NAT DHCP server provides a boot file name of the form `vmname.pxe` if the directory `TFTP` exists in the directory where the user's `VirtualBox.xml` file is kept. It is the responsibility of the user to provide `vmname.pxe`.

6.3.3 NAT limitations

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

ICMP protocol limitations: Some frequently used network debugging tools (e.g. `ping` or tracerouting) rely on the ICMP protocol for sending/receiving messages. While ICMP support has been improved with VirtualBox 2.1 (`ping` should now work), some other tools may not work reliably.

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 does not always work (but WINS always works). As a workaround, you can use the numeric IP of the desired server in the `\\server\share` notation.

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, Mac OS X) it is not possible to bind to ports below 1024 from applications that are not run by `root`. As a result, if you try to configure such a port forwarding, the VM will refuse to start.

These limitations normally 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).

6.4 Network Address Translation Service

The Network Address Translation (NAT) service works in a similar way to a home router, grouping the systems using it into a network and preventing systems outside of this network from directly accessing systems inside it, but letting systems inside communicate with each other and with systems outside using TCP and UDP over IPv4 and IPv6.

A NAT service is attached to an internal network. Virtual machines which are to make use of it should be attached to that internal network. The name of internal network is chosen when the NAT service is created and the internal network will be created if it does not already exist. An example command to create a NAT network is:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable
```

Here, "natnet1" is the name of the internal network to be used and "192.168.15.0/24" is the network address and mask of the NAT service interface. By default in this static configuration the gateway will be assigned the address 192.168.15.1 (the address following the interface address), though this is subject to change. To attach a DHCP server to the internal network, we modify the example as follows:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable --dhcp on
```

or to add a DHCP server to the network after creation:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp on
```

To disable it again, use:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp off
```

DHCP server provides list of registered nameservers, but doesn't map servers from 127/8 network.

To start the NAT service, use the following command:

```
VBoxManage natnetwork start --netname natnet1
```

If the network has a DHCP server attached then it will start together with the NAT network service.

```
VBoxManage natnetwork stop --netname natnet1
```

stops the NAT network service, together with DHCP server if any.

To delete the NAT network service use:

```
VBoxManage natnetwork remove --netname natnet1
```

This command does not remove the DHCP server if one is enabled on the internal network.

Port-forwarding is supported (using the `--port-forward-4` switch for IPv4 and `--port-forward-6` for IPv6):

```
VBoxManage natnetwork modify --netname natnet1 --port-forward-4 "ssh:tcp:[]:1022:[192.168.15.5]:22"
```

This adds a port-forwarding rule from the host's TCP 1022 port to the port 22 on the guest with IP address 192.168.15.5. Host port, guest port and guest IP are mandatory. To delete the rule, use:

```
VBoxManage natnetwork modify --netname natnet1 --port-forward-4 delete ssh
```

It's possible to bind NAT service to specified interface:

```
VBoxManage setextradata global "NAT/win-nat-test-0/SourceIp4" 192.168.1.185
```

To see the list of registered NAT networks, use:

```
VBoxManage list natnetworks
```

6.5 Bridged networking

With bridged networking, VirtualBox uses a device driver on your *host* system that filters data from your physical network adapter. This driver is therefore called a "net filter" driver. This allows VirtualBox to intercept data from the physical network and inject data into it, effectively creating a new network interface in software. When a guest is using such a new software interface, it looks to the host system as though the guest were physically connected to the interface using a network cable: the host can send data to the guest through that interface and receive data from it. This means that you can set up routing or bridging between the guest and the rest of your network.

For this to work, VirtualBox needs a device driver on your host system. The way bridged networking works has been completely rewritten with VirtualBox 2.0 and 2.1, depending on the host operating system. From the user perspective, the main difference is that complex configuration is no longer necessary on any of the supported host operating systems.[2]

> **Note:** Even though TAP is no longer necessary on Linux with bridged networking, you *can* still use TAP interfaces for certain advanced setups, since you can connect a VM to any host interface – which could also be a TAP interface.

To enable bridged networking, all you need to do is to open the Settings dialog of a virtual machine, go to the "Network" page and select "Bridged network" in the drop down list for the "Attached to" field. Finally, select desired host interface from the list at the bottom of the page, which contains the physical network interfaces of your systems. On a typical MacBook, for example, this will allow you to select between "en1: AirPort" (which is the wireless interface) and "en0: Ethernet", which represents the interface with a network cable.

> **Note:** Bridging to a wireless interface is done differently from bridging to a wired interface, because most wireless adapters do not support promiscuous mode. All traffic has to use the MAC address of the host's wireless adapter, and therefore VirtualBox needs to replace the source MAC address in the Ethernet header of an outgoing packet to make sure the reply will be sent to the host interface. When VirtualBox sees an incoming packet with a destination IP address that belongs to one of the virtual machine adapters it replaces the destination MAC address in the Ethernet header with the VM adapter's MAC address and passes it on. VirtualBox examines ARP and DHCP packets in order to learn the IP addresses of virtual machines.

Depending on your host operating system, the following limitations should be kept in mind:

- On **Macintosh** hosts, functionality is limited when using AirPort (the Mac's wireless networking) for bridged networking. Currently, VirtualBox supports only IPv4 and IPv6 over AirPort. For other protocols (such as IPX), you must choose a wired interface.
- On **Linux** hosts, functionality is limited when using wireless interfaces for bridged networking. Currently, VirtualBox supports only IPv4 and IPv6 over wireless. For other protocols (such as IPX), you must choose a wired interface.

  Also, setting the MTU to less than 1500 bytes on wired interfaces provided by the sky2 driver on the Marvell Yukon II EC Ultra Ethernet NIC is known to cause packet losses under certain conditions.

[2] For Mac OS X and Solaris hosts, net filter drivers were already added in VirtualBox 2.0 (as initial support for Host Interface Networking on these platforms). With VirtualBox 2.1, net filter drivers were also added for the Windows and Linux hosts, replacing the mechanisms previously present in VirtualBox for those platforms; especially on Linux, the earlier method required creating TAP interfaces and bridges, which was complex and varied from one distribution to the next. None of this is necessary anymore. Bridged network was formerly called "Host Interface Networking" and has been renamed with version 2.2 without any change in functionality.

Some adapters strip VLAN tags in hardware. This does not allow to use VLAN trunking between VM and the external network with pre-2.6.27 Linux kernels nor with host operating systems other than Linux.

- On **Solaris** hosts, there is no support for using wireless interfaces. Filtering guest traffic using IPFilter is also not completely supported due to technical restrictions of the Solaris networking subsystem. These issues would be addressed in a future release of Solaris 11.

  Starting with VirtualBox 4.1, on Solaris 11 hosts (build 159 and above), it is possible to use Solaris' Crossbow Virtual Network Interfaces (VNICs) directly with VirtualBox without any additional configuration other than each VNIC must be exclusive for every guest network interface.

  Starting with VirtualBox 2.0.4 and up to VirtualBox 4.0, VNICs can be used but with the following caveats:

  - A VNIC cannot be shared between multiple guest network interfaces, i.e. each guest network interface must have its own, exclusive VNIC.
  - The VNIC and the guest network interface that uses the VNIC must be assigned identical MAC addresses.

  When using VLAN interfaces with VirtualBox, they must be named according to the PPA-hack naming scheme (e.g. "e1000g513001"), as otherwise the guest may receive packets in an unexpected format.

6.6 Internal networking

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

Even though technically, everything that can be done using internal networking can also be done using bridged networking, there are security advantages with internal networking. In bridged networking mode, all traffic goes through a physical interface of the host system. It is therefore possible to attach a packet sniffer (such as Wireshark) to the host interface and log all traffic that goes over it. If, for any reason, you prefer two or more VMs on the same machine to communicate privately, hiding their data from both the host system and the user, bridged networking therefore is not an option.

Internal networks are created automatically as needed, 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.

Unless you configure the (virtual) network cards in the guest operating systems that are participating in the internal network to use static IP addresses, you may want to use the DHCP server that is built into VirtualBox to manage IP addresses for the internal network. Please see chapter 8.38, *VBoxManage dhcpserver*, page 176 for details.

As a security measure, by default, the Linux implementation of internal networking only allows VMs running under the same user ID to establish an internal network. However, it is possible to create a shared internal networking interface, accessible by users with different UUIds.

6.7 Host-only networking

Host-only networking is another networking mode that was added with version 2.2 of VirtualBox. It can be thought of as a hybrid between the bridged and internal networking modes: as with bridged networking, the virtual machines can talk to each other and the host as if they were connected through a physical Ethernet switch. Similarly, as with internal networking however, a physical networking interface need not be present, and the virtual machines cannot talk to the world outside the host since they are not connected to a physical networking interface.

Instead, when host-only networking is used, VirtualBox creates a new software interface on the host which then appears next to your existing network interfaces. In other words, whereas with bridged networking an existing physical interface is used to attach virtual machines to, with host-only networking a new "loopback" interface is created on the host. And whereas with internal networking, the traffic between the virtual machines cannot be seen, the traffic on the "loopback" interface on the host can be intercepted.

Host-only networking is particularly useful for preconfigured virtual appliances, where multiple virtual machines are shipped together and designed to cooperate. For example, one virtual machine may contain a web server and a second one a database, and since they are intended to talk to each other, the appliance can instruct VirtualBox to set up a host-only network for the two. A second (bridged) network would then connect the web server to the outside world to serve data to, but the outside world cannot connect to the database.

To change a virtual machine's virtual network interface to "host only" mode:

- either go to the "Network" page in the virtual machine's settings notebook in the graphical user interface and select "Host-only networking", or
- on the command line, type `VBoxManage modifyvm "VM name" --nic<x> hostonly`; see chapter 8.8, *VBoxManage modifyvm*, page 128 for details.

Before you can attach a VM to a host-only network you have to create at least one host-only interface, either from the GUI: "File" → "Preferences" → "Network" → "Host-only network" → "(+)Add host-only network", or via command line with

```
VBoxManage hostonlyif create
```

see chapter 8.37, *VBoxManage hostonlyif*, page 176 for details.

For host-only networking, like with internal networking, you may find the DHCP server useful that is built into VirtualBox. This can be enabled to then manage the IP addresses in the host-only network since otherwise you would need to configure all IP addresses statically.

- In the VirtualBox graphical user interface, you can configure all these items in the global settings via "File" → "Preferences" → "Network", which lists all host-only networks which are presently in use. Click on the network name and then on the "Edit" button to the right, and you can modify the adapter and DHCP settings.

- Alternatively, you can use `VBoxManage dhcpserver` on the command line; please see chapter 8.38, *VBoxManage dhcpserver*, page 176 for details.

> **Note:** On Linux and Mac OS X hosts the number of host-only interfaces is limited to 128. There is no such limit for Solaris and Windows hosts.

6.8 UDP Tunnel networking

This networking mode allows to interconnect virtual machines running on different hosts.

Technically this is done by encapsulating Ethernet frames sent or received by the guest network card into UDP/IP datagrams, and sending them over any network available to the host.

UDP Tunnel mode has three parameters:

Source UDP port The port on which the host listens. Datagrams arriving on this port from any source address will be forwarded to the receiving part of the guest network card.

Destination address IP address of the target host of the transmitted data.

Destination UDP port Port number to which the transmitted data is sent.

When interconnecting two virtual machines on two different hosts, their IP addresses must be swapped. On single host, source and destination UDP ports must be swapped.

In the following example host 1 uses the IP address 10.0.0.1 and host 2 uses IP address 10.0.0.2. Configuration via command-line:

```
VBoxManage modifyvm "VM 01 on host 1" --nic<x> generic
VBoxManage modifyvm "VM 01 on host 1" --nicgenericdrv<x> UDPTunnel
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> dest=10.0.0.2
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> sport=10001
VBoxManage modifyvm "VM 01 on host 1" --nicproperty<x> dport=10002
```

and

```
VBoxManage modifyvm "VM 02 on host 2" --nic<y> generic
VBoxManage modifyvm "VM 02 on host 2" --nicgenericdrv<y> UDPTunnel
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> dest=10.0.0.1
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> sport=10002
VBoxManage modifyvm "VM 02 on host 2" --nicproperty<y> dport=10001
```

Of course, you can always interconnect two virtual machines on the same host, by setting the destination address parameter to 127.0.0.1 on both. It will act similarly to “Internal network” in this case, however the host can see the network traffic which it could not in the normal Internal network case.

> **Note:** On Unix-based hosts (e.g. Linux, Solaris, Mac OS X) it is not possible to bind to ports below 1024 from applications that are not run by `root`. As a result, if you try to configure such a source UDP port, the VM will refuse to start.

6.9 VDE networking

Virtual Distributed Ethernet (VDE[3]) is a flexible, virtual network infrastructure system, spanning across multiple hosts in a secure way. It allows for L2/L3 switching, including spanning-tree protocol, VLANs, and WAN emulation. It is an optional part of VirtualBox which is only included in the source code.

The basic building blocks of the infrastructure are VDE switches, VDE plugs and VDE wires which inter-connect the switches.

The VirtualBox VDE driver has one parameter:

VDE network The name of the VDE network switch socket to which the VM will be connected.

The following basic example shows how to connect a virtual machine to a VDE switch:

1. Create a VDE switch:

   ```
   vde_switch -s /tmp/switch1
   ```

2. Configuration via command-line:

   ```
   VBoxManage modifyvm "VM name" --nic<x> generic
   ```

   ```
   VBoxManage modifyvm "VM name" --nicgenericdrv<x> VDE
   ```

   To connect to automatically allocated switch port, use:

   ```
   VBoxManage modifyvm "VM name" --nicproperty<x> network=/tmp/switch1
   ```

   To connect to specific switch port <n>, use:

   ```
   VBoxManage modifyvm "VM name" --nicproperty<x> network=/tmp/switch1[<n>]
   ```

   The latter option can be useful for VLANs.

3. Optionally map between VDE switch port and VLAN: (from switch CLI)

   ```
   vde$ vlan/create <VLAN>
   ```

   ```
   vde$ port/setvlan <port> <VLAN>
   ```

VDE is available on Linux and FreeBSD hosts only. It is only available if the VDE software and the VDE plugin library from the VirtualSquare project are installed on the host system[4]. For more information on setting up VDE networks, please see the documentation accompanying the software.[5]

6.10 Limiting bandwidth for network I/O

Starting with version 4.2, VirtualBox allows for limiting the maximum bandwidth used for network transmission. Several network adapters of one VM may share limits through bandwidth groups. It is possible to have more than one such limit.

> **Note:** VirtualBox shapes VM traffic only in the transmit direction, delaying the packets being sent by virtual machines. It does not limit the traffic being received by virtual machines.

Limits are configured through `VBoxManage`. The example below creates a bandwidth group named "Limit", sets the limit to 20 Mbit/s and assigns the group to the first and second adapters of the VM:

[3] VDE is a project developed by Renzo Davoli, Associate Professor at the University of Bologna, Italy.
[4] For Linux hosts, the shared library libvdeplug.so must be available in the search path for shared libraries
[5] http://wiki.virtualsquare.org/wiki/index.php/VDE_Basic_Networking.

```
VBoxManage bandwidthctl "VM name" add Limit --type network --limit 20m
VBoxManage modifyvm "VM name" --nicbandwidthgroup1 Limit
VBoxManage modifyvm "VM name" --nicbandwidthgroup2 Limit
```

All adapters in a group share the bandwidth limit, meaning that in the example above the bandwidth of both adapters combined can never exceed 20 Mbit/s. However, if one adapter doesn't require bandwidth the other can use the remaining bandwidth of its group.

The limits for each group can be changed while the VM is running, with changes being picked up immediately. The example below changes the limit for the group created in the example above to 100 Kbit/s:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 100k
```

To completely disable shaping for the first adapter of VM use the following command:

```
VBoxManage modifyvm "VM name" --nicbandwidthgroup1 none
```

It is also possible to disable shaping for all adapters assigned to a bandwidth group while VM is running, by specifying the zero limit for the group. For example, for the bandwidth group named "Limit" use:

```
VBoxManage bandwidthctl "VM name" set Limit --limit 0
```

6.11 Improving network performance

VirtualBox provides a variety of virtual network adapters that can be "attached" to the host's network in a number of ways. Depending on which types of adapters and attachments are used the network performance will be different. Performance-wise the *virtio* network adapter is preferable over *Intel PRO/1000* emulated adapters, which are preferred over *PCNet* family of adapters. Both *virtio* and *Intel PRO/1000* adapters enjoy the benefit of segmentation and checksum offloading. Segmentation offloading is essential for high performance as it allows for less context switches, dramatically increasing the sizes of packets that cross VM/host boundary.

> **Note:** Neither *virtio* nor *Intel PRO/1000* drivers for Windows XP support segmentation offloading. Therefore Windows XP guests never reach the same transmission rates as other guest types. Refer to MS Knowledge base article 842264 for additional information.

Three attachment types: *internal*, *bridged* and *host-only*, have nearly identical performance, the *internal* type being a little bit faster and using less CPU cycles as the packets never reach the host's network stack. The *NAT* attachment is the slowest (and safest) of all attachment types as it provides network address translation. The generic driver attachment is special and cannot be considered as an alternative to other attachment types.

The number of CPUs assigned to VM does not improve network performance and in some cases may hurt it due to increased concurrency in the guest.

Here is the short summary of things to check in order to improve network performance:

1. Whenever possible use *virtio* network adapter, otherwise use one of *Intel PRO/1000* adapters;
2. Use *bridged* attachment instead of *NAT*;
3. Make sure segmentation offloading is enabled in the guest OS. Usually it will be enabled by default. You can check and modify offloading settings using `ethtool` command in Linux guests.

4. Perform a full, detailed analysis of network traffic on the VM's network adaptor using a 3rd party tool such as Wireshark. To do this, a promiscuous mode policy needs to be used on the VM's network adaptor. Use of this mode is only possible on networks: NAT Network, Bridged Adapter, Internal Network and Host-only Adapter.

   To setup a promiscuous mode policy, either select from the drop down list located in the Network Settings dialog for the network adaptor or use the command line tool `VBoxManage`; for details, refer to chapter 8.8, *VBoxManage modifyvm*, page 128.

   Promiscuous mode policies are:

   a) `deny` (default setting) which hides any traffic not intended for this VM's network adaptor.
   b) `allow-vms` which hides all host traffic from this VM's network adaptor, but allows it to see traffic from/to other VMs.
   c) `allow-all` which removes all restrictions - this VM's network adaptor sees all traffic.

7 Remote virtual machines

7.1 Remote display (VRDP support)

VirtualBox can display virtual machines remotely, meaning that a virtual machine can execute on one computer even though the machine will be displayed on a second computer, and the machine will be controlled from there as well, as if the virtual machine was running on that second computer.

For maximum flexibility, starting with VirtualBox 4.0, VirtualBox implements remote machine display through a generic extension interface, the VirtualBox Remote Desktop Extension (VRDE). The base open-source VirtualBox package only provides this interface, while implementations can be supplied by third parties with VirtualBox extension packages, which must be installed separately from the base package. See chapter 1.5, *Installing VirtualBox and extension packs*, page 16 for more information.

Oracle provides support for the **VirtualBox Remote Display Protocol (VRDP)** in such a VirtualBox extension package. When this package is installed, VirtualBox versions 4.0 and later support VRDP the same way as binary (non-open-source) versions of VirtualBox before 4.0 did.

VRDP is a backwards-compatible extension to Microsoft's Remote Desktop Protocol (RDP). As a result, you can use any standard RDP client to control the remote VM.

Even when the extension is installed, the VRDP server is disabled by default. It can easily be enabled on a per-VM basis either in the VirtualBox Manager in the "Display" settings (see chapter 3.5, *Display settings*, page 51) or with `VBoxManage`:

```
VBoxManage modifyvm "VM name" --vrde on
```

By default, the VRDP server uses TCP port `3389`. You will need to change the default port if you run more than one VRDP server, since the port can only be used by one server at a time; you might also need to change it on Windows hosts since the default port might already be used by the RDP server that is built into Windows itself. Ports 5000 through 5050 are typically not used and might be a good choice.

The port can be changed either in the "Display" settings of the graphical user interface or with `--vrdeport` option of the `VBoxManage modifyvm` command. You can specify a comma-separated list of ports or ranges of ports. Use a dash between two port numbers to specify a range. The VRDP server will bind to **one** of available ports from the specified list. For example, `VBoxManage modifyvm "VM name" --vrdeport 5000,5010-5012` will configure the server to bind to one of the ports 5000, 5010, 5011 or 5012. See chapter 8.8.5, *Remote machine settings*, page 136 for details.

The actual port used by a running VM can be either queried with `VBoxManage showvminfo` command or seen in the GUI on the "Runtime" tab of the "Session Information Dialog", which is accessible via the "Machine" menu of the VM window.

Support for IPv6 has been implemented in VirtualBox 4.3. If the host OS supports IPv6 the VRDP server will automatically listen for IPv6 connections in addition to IPv4.

7.1.1 Common third-party RDP viewers

Since VRDP is backwards-compatible to RDP, you can use any standard RDP viewer to connect to such a remote virtual machine (examples follow below). For this to work, you must specify the **IP address** of your *host* system (not of the virtual machine!) as the server address to connect to, as well as the **port number** that the VRDP server is using.

Here follow examples for the most common RDP viewers:

- On Windows, you can use the Microsoft Terminal Services Connector (`mstsc.exe`) that ships with Windows. You can start it by bringing up the "Run" dialog (press the Windows key and "R") and typing "mstsc". You can also find it under "Start" → "All Programs" → "Accessories" → "Remote Desktop Connection". If you use the "Run" dialog, you can type in options directly:

  ```
  mstsc 1.2.3.4:3389
  ```

  Replace `1.2.3.4` with the host IP address, and `3389` with a different port if necessary.

  > **Note:** IPv6 address must be enclosed in square brackets to specify a port. For example: `mstsc [fe80::1:2:3:4]:3389`

  > **Note:** When connecting to localhost in order to test the connection, the addresses `localhost` and `127.0.0.1` might not work using `mstsc.exe`. Instead, the address `127.0.0.2[:3389]` has to be used.

- On other systems, you can use the standard open-source `rdesktop` program. This ships with most Linux distributions, but VirtualBox also comes with a modified variant of rdesktop for remote USB support (see chapter 7.1.4, *Remote USB*, page 108 below).

  With rdesktop, use a command line such as the following:

  ```
  rdesktop -a 16 -N 1.2.3.4:3389
  ```

  As said for the Microsoft viewer above, replace `1.2.3.4` with the host IP address, and `3389` with a different port if necessary. The `-a 16` option requests a color depth of 16 bits per pixel, which we recommend. (For best performance, after installation of the guest operating system, you should set its display color depth to the same value). The `-N` option enables use of the NumPad keys.

- If you run the KDE desktop, you might prefer `krdc`, the KDE RDP viewer. The command line would look like this:

  ```
  krdc rdp://1.2.3.4:3389
  ```

  Again, replace `1.2.3.4` with the host IP address, and `3389` with a different port if necessary. The "rdp://" bit is required with krdc to switch it into RDP mode.

- With Sun Ray thin clients you can use `uttsc`, which is part of the Sun Ray Windows Connector package. See the corresponding documentation for details.

7.1.2 VBoxHeadless, the remote desktop server

While any VM started from the VirtualBox Manager is capable of running virtual machines remotely, it is not convenient to have to run the full-fledged GUI if you never want to have VMs displayed locally in the first place. In particular, if you are running server hardware 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, the VirtualBox manager comes with dependencies on the Qt and SDL libraries. This 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 called `VBoxHeadless`, which produces no visible output on the host at all, but still can deliver VRDP data. This front-end has no dependencies on the X Window system on Linux and Solaris hosts.[1]

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

- You can use

  ```
  VBoxManage startvm "VM name" --type headless
  ```

  The extra `--type` option causes VirtualBox to use `VBoxHeadless` as the front-end to the internal virtualization engine instead of the Qt front-end.

- One alternative is to use `VBoxHeadless` directly, as follows:

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

  This way of starting the VM helps troubleshooting problems reported by `VBoxManage startvm ...` because you can see sometimes more detailed error messages, especially for early failures before the VM execution is started. In normal situations `VBoxManage startvm` is preferred since it runs the VM directly as a background process which has to be done explicitly when directly starting `VBoxHeadless`.

- The other alternative is to start `VBoxHeadless` from the VirtualBox Manager GUI, by holding the Shift key when starting a virtual machine or selecting `Headless Start` from the `Machine` menu.

Since VirtualBox version 5.0, when you use `VBoxHeadless` to start a VM, the VRDP server will be enabled according to the VM configuration. You can override the VM's setting using `--vrde` command line parameter. To enable the VRDP server start the VM like this:

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

and to disable it:

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

To have the VRDP server enabled depending on the VM configuration, as the other front-ends would, you can still use:

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

but this is the same as

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

If you start the VM with `VBoxManage startvm ...` then the configuration settings of the VM are always used.

7.1.3 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 an RDP 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. The VirtualBox extension pack for the VRDP server must be installed (see the previous section). For the following example, we will assume a Linux server.

[1] Before VirtualBox 1.6, the headless server was called `VBoxVRDP`. For the sake of backwards compatibility, the VirtualBox installation still installs an executable with that name as well.

2. An ISO file accessible from 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 through which you can access a command line (e.g. via `ssh`).

4. An RDP viewer on the remote client; see chapter 7.1.1, *Common third-party RDP viewers*, page 104 above for examples.

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" --ostype WindowsXP --register
   ```

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

   Note further that you do not need to specify `--ostype`, but doing so selects some sane default values for certain VM parameters, for example the RAM size and the type of the virtual network device. To get a complete list of supported operating systems you can use

   ```
   VBoxManage list ostypes
   ```

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 256 --acpi on --boot1 dvd --nic1 nat
   ```

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

   ```
   VBoxManage createhd --filename "WinXP.vdi" --size 10000
   ```

4. Add an IDE Controller to the new VM:

   ```
   VBoxManage storagectl "Windows XP" --name "IDE Controller"
         --add ide --controller PIIX4
   ```

5. Set the VDI file created above as the first virtual hard disk of the new VM:

   ```
   VBoxManage storageattach "Windows XP" --storagectl "IDE Controller"
         --port 0 --device 0 --type hdd --medium "WinXP.vdi"
   ```

6. Attach the ISO file that contains the operating system installation that you want to install later to the virtual machine, so the machine can boot from it:

   ```
   VBoxManage storageattach "Windows XP" --storagectl "IDE Controller"
         --port 0 --device 1 --type dvddrive --medium /full/path/to/iso.iso
   ```

7. Enable VirtualBox remote desktop extension (the VRDP server):

   ```
   VBoxManage modifyvm "Windows XP" --vrde on
   ```

8. 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.

9. On the client machine, fire up the RDP viewer and try to connect to the server (see chapter 7.1.1, *Common third-party RDP viewers*, page 104 above for how to use various common RDP viewers).

   You should now be seeing the installation routine of your guest operating system remotely in the RDP viewer.

7.1.4 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 VRDP 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 3.10.1, *USB settings*, page 56. All you have to do is specify "Remote" (or "Any") when setting up these rules.

Accessing remote USB devices is only possible if the RDP client supports this extension. On Linux and Solaris hosts, the VirtualBox installation provides a suitable VRDP client called `rdesktop-vrdp`. Recent versions of `uttsc`, a client tailored for the use with Sun Ray thin clients, also support accessing remote USB devices. RDP clients for other platforms will be provided in future VirtualBox versions.

To make a remote USB device available to a VM, `rdesktop-vrdp` should be started as follows:

```
rdesktop-vrdp -r usb -a 16 -N my.host.address
```

Please refer to chapter 12.8.7, *USB not working*, page 262 for further details on how to properly set up the permissions for USB devices. Furthermore it is advisable to disable automatic loading of any host driver on the remote host which might work on USB devices to ensure that the devices are accessible by the RDP client. If the setup was properly done on the remote host, plug/unplug events are visible on the VBox.log file of the VM.

7.1.5 RDP authentication

For each virtual machine that is remotely accessible via RDP, you can individually determine if and how client connections are authenticated. For this, use `VBoxManage modifyvm` command with the `--vrdeauthtype` option; see chapter 8.8, *VBoxManage modifyvm*, page 128 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 ships with two such authentication libraries:
  1. The default authentication library, `VBoxAuth`, authenticates against user credentials of the hosts. Depending on the host platform, this means:
     - On Linux hosts, `VBoxAuth.so` authenticates users against the host's PAM system.
     - On Windows hosts, `VBoxAuth.dll` authenticates users against the host's WinLogon system.
     - On Mac OS X hosts, `VBoxAuth.dylib` authenticates users against the host's directory service.[2]

     In other words, the "external" method per default performs authentication with the user accounts that exist on the host system. Any user with valid authentication credentials is accepted, i.e. the username does not have to correspond to the user running the VM.

[2] Support for Mac OS X was added in version 3.2.

2. An additional library called `VBoxAuthSimple` performs authentication against credentials configured in the "extradata" section of a virtual machine's XML settings file. This is probably the simplest way to get authentication that does not depend on a running and supported guest (see below). The following steps are required:

   a) Enable `VBoxAuthSimple` with the following command:

   ```
   VBoxManage setproperty vrdeauthlibrary "VBoxAuthSimple"
   ```

   b) To enable the library for a particular VM, you must then switch authentication to external:

   ```
   VBoxManage modifyvm "VM name" --vrdeauthtype external
   ```

   Replace `<vm>` with the VM name or UUID.

   c) You will then need to configure users and passwords by writing items into the machine's extradata. Since the XML machine settings file, into whose "extradata" section the password needs to be written, is a plain text file, VirtualBox uses hashes to encrypt passwords. The following command must be used:

   ```
   VBoxManage setextradata "VM name" "VBoxAuthSimple/users/<user>" <hash>
   ```

   Replace `<vm>` with the VM name or UUID, `<user>` with the user name who should be allowed to log in and `<hash>` with the encrypted password. As an example, to obtain the hash value for the password "secret", you can use the following command:

   ```
   VBoxManage internalcommands passwordhash "secret"
   ```

   This will print

   ```
   2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b
   ```

   You can then use VBoxManage setextradata to store this value in the machine's "extradata" section.

   As example, combined together, to set the password for the user "john" and the machine "My VM" to "secret", use this command:

   ```
   VBoxManage setextradata "My VM" "VBoxAuthSimple/users/john"
       2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b
   ```

- Finally, the "guest" authentication method performs authentication with a special component that comes with the Guest Additions; as a result, authentication is not performed on the host, but with the *guest* user accounts.

  This method is currently still in testing and not yet supported.

In addition to the methods described above, 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. This is described in detail in the VirtualBox Software Development Kit (SDK) reference; please see chapter 11, *VirtualBox programming interfaces*, page 245 for details.

7.1.6 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 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.

3. RDP5.2 authentication uses the Enhanced RDP Security, which means that an external security protocol is used to secure the connection. RDP4 and RDP5.1 use Standard RDP Security. The VRDP server supports Enhanced RDP Security with TLS protocol and, as a part of TLS handshake, sends the server certificate to the client.

   The `Security/Method` VRDE property sets the desired security method, which is used for a connection. Valid values are:

   - `Negotiate` - both Enhanced (TLS) and Standard RDP Security connections are allowed. The security method is negotiated with the client. This is the default setting.
   - `RDP` - only Standard RDP Security is accepted.
   - `TLS` - only Enhanced RDP Security is accepted. The client must support TLS.

   For example the following command allows a client to use either Standard or Enhanced RDP Security connection:

   ```
   vboxmanage modifyvm "VM name" --vrdeproperty "Security/Method=negotiate"
   ```

   If the `Security/Method` property is set to either `Negotiate` or `TLS`, the TLS protocol will be automatically used by the server, if the client supports TLS. However, in order to use TLS the server must possess the Server Certificate, the Server Private Key and the Certificate Authority (CA) Certificate. The following example shows how to generate a server certificate.

   a) Create a CA self signed certificate:

   ```
   openssl req -new -x509 -days 365 -extensions v3_ca \
     -keyout ca_key_private.pem -out ca_cert.pem
   ```

   b) Generate a server private key and a request for signing:

   ```
   openssl genrsa -out server_key_private.pem
   openssl req -new -key server_key_private.pem -out server_req.pem
   ```

   c) Generate the server certificate:

   ```
   openssl x509 -req -days 365 -in server_req.pem \
     -CA ca_cert.pem -CAkey ca_key_private.pem -set_serial 01 -out server_cert.pem
   ```

   The server must be configured to access the required files:

   ```
   vboxmanage modifyvm "VM name" \
     --vrdeproperty "Security/CACertificate=path/ca_cert.pem"

   vboxmanage modifyvm "VM name" \
     --vrdeproperty "Security/ServerCertificate=path/server_cert.pem"

   vboxmanage modifyvm "VM name" \
     --vrdeproperty "Security/ServerPrivateKey=path/server_key_private.pem"
   ```

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.

7.1.7 Multiple connections to the VRDP server

The VRDP server of VirtualBox supports multiple 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 "VM name" --vrdemulticon on
```

7.1.8 Multiple remote monitors

To access two or more remote VM displays you have to enable the VRDP multiconnection mode (see chapter 7.1.7, *Multiple connections to the VRDP server*, page 110).

The RDP client can select the virtual monitor number to connect to using the `domain` logon parameter (`-d`). If the parameter ends with `@` followed by a number, VirtualBox interprets this number as the screen index. The primary guest screen is selected with `@1`, the first secondary screen is `@2`, etc.

The Microsoft 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.

7.1.9 VRDP video redirection

Starting with VirtualBox 3.2, the VRDP server can redirect video streams from the guest to the RDP client. Video frames are compressed using the JPEG algorithm allowing a higher compression ratio than standard RDP bitmap compression methods. It is possible to increase the compression ratio by lowering the video quality.

The VRDP server automatically detects video streams in a guest as frequently updated rectangular areas. As a result, this method works with any guest operating system without having to install additional software in the guest; in particular, the Guest Additions are not required.

On the client side, however, currently only the Windows 7 Remote Desktop Connection client supports this feature. If a client does not support video redirection, the VRDP server falls back to regular bitmap updates.

The following command enables video redirection:

```
VBoxManage modifyvm "VM name" --vrdevideochannel on
```

The quality of the video is defined as a value from 10 to 100 percent, representing a JPEG compression level (where lower numbers mean lower quality but higher compression). The quality can be changed using the following command:

```
VBoxManage modifyvm "VM name" --vrdevideochannelquality 75
```

7.1.10 VRDP customization

With VirtualBox 4.0 it is possible to disable display output, mouse and keyboard input, audio, remote USB or clipboard individually in the VRDP server.

The following commands change corresponding server settings:

```
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableDisplay=1
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableInput=1
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableUSB=1
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableAudio=1
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableClipboard=1
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableUpstreamAudio=1
```

To reenable a feature use a similar command without the trailing 1. For example:

```
VBoxManage modifyvm "VM name" --vrdeproperty Client/DisableDisplay=
```

These properties were introduced with VirtualBox 3.2.10. However, in the 3.2.x series, it was necessary to use the following commands to alter these settings instead:

```
VBoxManage setextradata "VM name" "VRDP/Feature/Client/DisableDisplay" 1
VBoxManage setextradata "VM name" "VRDP/Feature/Client/DisableInput" 1
VBoxManage setextradata "VM name" "VRDP/Feature/Client/DisableUSB" 1
VBoxManage setextradata "VM name" "VRDP/Feature/Client/DisableAudio" 1
VBoxManage setextradata "VM name" "VRDP/Feature/Client/DisableClipboard" 1
```

To reenable a feature use a similar command without the trailing 1. For example:

```
VBoxManage setextradata "VM name" "VRDP/Feature/Client/DisableDisplay"
```

7.2 Teleporting

Starting with version 3.1, VirtualBox supports "teleporting" – that is, moving a virtual machine over a network from one VirtualBox host to another, while the virtual machine is running. This works regardless of the host operating system that is running on the hosts: you can teleport virtual machines between Solaris and Mac hosts, for example.

Teleporting requires that a machine be currently running on one host, which is then called the **"source"**. The host to which the virtual machine will be teleported will then be called the **"target"**; the machine on the target is then configured to wait for the source to contact the target. The machine's running state will then be transferred from the source to the target with minimal downtime.

Teleporting happens over any TCP/IP network; the source and the target only need to agree on a TCP/IP port which is specified in the teleporting settings.

At this time, there are a few prerequisites for this to work, however:

1. On the target host, you must configure a virtual machine in VirtualBox with exactly the same hardware settings as the machine on the source that you want to teleport. This does not apply to settings which are merely descriptive, such as the VM name, but obviously for teleporting to work, the target machine must have the same amount of memory and other hardware settings. Otherwise teleporting will fail with an error message.

2. The two virtual machines on the source and the target must share the same storage (hard disks as well as floppy and CD/DVD images). This means that they either use the same iSCSI targets or that the storage resides somewhere on the network and both hosts have access to it via NFS or SMB/CIFS.

   This also means that neither the source nor the target machine can have any snapshots.

Then perform the following steps:

1. On the *target* host, configure the virtual machine to wait for a teleport request to arrive when it is started, instead of actually attempting to start the machine. This is done with the following VBoxManage command:

   ```
   VBoxManage modifyvm <targetvmname> --teleporter on --teleporterport <port>
   ```

   where `<targetvmname>` is the name of the virtual machine on the target host and `<port>` is a TCP/IP port number to be used on both the source and the target hosts. For example, use 6000. For details, see chapter 8.8.6, *Teleporting settings*, page 139.

2. Start the VM on the target host. You will see that instead of actually running, it will show a progress dialog. indicating that it is waiting for a teleport request to arrive.

3. Start the machine on the *source* host as usual. When it is running and you want it to be teleported, issue the following command on the source host:

   ```
   VBoxManage controlvm <sourcevmname> teleport --host <targethost> --port <port>
   ```

   where `<sourcevmname>` is the name of the virtual machine on the source host (the machine that is currently running), `<targethost>` is the host or IP name of the target host on which the machine is waiting for the teleport request, and `<port>` must be the same number as specified in the command on the target host. For details, see chapter 8.13, *VBoxManage controlvm*, page 143.

For testing, you can also teleport machines on the same host; in that case, use “localhost” as the hostname on both the source and the target host.

Note: In rare cases, if the CPUs of the source and the target are very different, teleporting can fail with an error message, or the target may hang. This may happen especially if the VM is running application software that is highly optimized to run on a particular CPU without correctly checking that certain CPU features are actually present. VirtualBox filters what CPU capabilities are presented to the guest operating system. Advanced users can attempt to restrict these virtual CPU capabilities with the `VBoxManage --modifyvm --cpuid` command; see chapter 8.8.6, *Teleporting settings*, page 139.

8 VBoxManage

8.1 Introduction

As briefly mentioned in chapter 1.16, *Alternative front-ends*, page 32, VBoxManage is the command-line interface to VirtualBox. With it, you can completely control VirtualBox from the command line of your host operating system. VBoxManage supports all the features that the graphical user interface gives you access to, but it supports a lot more than that. It exposes 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" or "createvm" or "startvm". All the subcommands that `VBoxManage` supports are described in detail in chapter 8, *VBoxManage*, page 114.

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:

  ```
  $ VBoxManage createvm --name "SUSE 10.2" --register
  VirtualBox Command Line Management Interface Version 5.1.22
  (C) 2005-2017 Oracle Corporation
  All rights reserved.

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

[1] For details, see chapter 8.7, *VBoxManage createvm*, page 127.

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 8.5, *VBoxManage showvminfo*, page 126 for details and an example.
- To change settings while a VM is powered off, use `VBoxManage modifyvm`, e.g. as follows:

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

  For details, see chapter 8.8, *VBoxManage modifyvm*, page 128.
- To change the storage configuration (e.g. to add a storage controller and then a virtual disk), use `VBoxManage storagectl` and `VBoxManage storageattach`; see chapter 8.19, *VBoxManage storagectl*, page 154 and chapter 8.18, *VBoxManage storageattach*, page 150 for details.
- To control VM operation, use one of the following:
  - To start a VM that is currently powered off, use `VBoxManage startvm`; see chapter 8.12, *VBoxManage startvm*, page 142 for details.
  - To pause or save a VM that is currently running or change some of its settings, use `VBoxManage controlvm`; see chapter 8.13, *VBoxManage controlvm*, page 143 for details.

8.2 Commands overview

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 [<general option>] <command>


General Options:

  [-v|--version]            print version number and exit
  [-q|--nologo]             suppress the logo
  [--settingspw <pw>]       provide the settings password
  [--settingspwfile <file>] provide a file containing the settings password


Commands:

  list [--long|-l]          vms|runningvms|ostypes|hostdvds|hostfloppies|
                            intnets|bridgedifs|hostonlyifs|natnets|dhcpservers|
                            hostinfo|hostcpuids|hddbackends|hdds|dvds|floppies|
                            usbhost|usbfilters|systemproperties|extpacks|
                            groups|webcams|screenshotformats

  showvminfo                <uuid|vmname> [--details]
                            [--machinereadable]
  showvminfo                <uuid|vmname> --log <idx>

  registervm                <filename>

  unregistervm              <uuid|vmname> [--delete]

  createvm                  --name <name>
```

```
                        [--groups <group>, ...]
                        [--ostype <ostype>]
                        [--register]
                        [--basefolder <path>]
                        [--uuid <uuid>]

modifyvm                <uuid|vmname>
                        [--name <name>]
                        [--groups <group>, ...]
                        [--description <desc>]
                        [--ostype <ostype>]
                        [--iconfile <filename>]
                        [--memory <memorysize in MB>]
                        [--pagefusion on|off]
                        [--vram <vramsize in MB>]
                        [--acpi on|off]
                        [--pciattach 03:04.0]
                        [--pciattach 03:04.0@02:01.0]
                        [--pcidetach 03:04.0]
                        [--ioapic on|off]
                        [--hpet on|off]
                        [--triplefaultreset on|off]
                        [--apic on|off]
                        [--x2apic on|off]
                        [--paravirtprovider none|default|legacy|minimal|
                                            hyperv|kvm]
                        [--paravirtdebug <key=value> [,<key=value> ...]]
                        [--hwvirtex on|off]
                        [--nestedpaging on|off]
                        [--largepages on|off]
                        [--vtxvpid on|off]
                        [--vtxux on|off]
                        [--pae on|off]
                        [--longmode on|off]
                        [--cpu-profile "host|Intel 80[86|286|386]"]
                        [--cpuid-portability-level <0..3>
                        [--cpuidset <leaf> <eax> <ebx> <ecx> <edx>]
                        [--cpuidremove <leaf>]
                        [--cpuidremoveall]
                        [--hardwareuuid <uuid>]
                        [--cpus <number>]
                        [--cpuhotplug on|off]
                        [--plugcpu <id>]
                        [--unplugcpu <id>]
                        [--cpuexecutioncap <1-100>]
                        [--rtcuseutc on|off]
                        [--graphicscontroller none|vboxvga|vmsvga]
                        [--monitorcount <number>]
                        [--accelerate3d on|off]
                        [--accelerate2dvideo on|off]
                        [--firmware bios|efi|efi32|efi64]
                        [--chipset ich9|piix3]
                        [--bioslogofadein on|off]
                        [--bioslogofadeout on|off]
                        [--bioslogodisplaytime <msec>]
                        [--bioslogoimagepath <imagepath>]
                        [--biosbootmenu disabled|menuonly|messageandmenu]
                        [--biosapic disabled|apic|x2apic]
                        [--biossystemtimeoffset <msec>]
                        [--biospxedebug on|off]
                        [--boot<1-4> none|floppy|dvd|disk|net>]
                        [--nic<1-N> none|null|nat|bridged|intnet|hostonly|
                                    generic|natnetwork]
                        [--nictype<1-N> Am79C970A|Am79C973|
                                        82540EM|82543GC|82545EM|
                                        virtio]
                        [--cableconnected<1-N> on|off]
```

```
[--nictrace<1-N> on|off]
[--nictracefile<1-N> <filename>]
[--nicproperty<1-N> name=[value]]
[--nicspeed<1-N> <kbps>]
[--nicbootprio<1-N> <priority>]
[--nicpromisc<1-N> deny|allow-vms|allow-all]
[--nicbandwidthgroup<1-N> none|<name>]
[--bridgeadapter<1-N> none|<devicename>]
[--hostonlyadapter<1-N> none|<devicename>]
[--intnet<1-N> <network name>]
[--nat-network<1-N> <network name>]
[--nicgenericdrv<1-N> <driver>
[--natnet<1-N> <network>|default]
[--natsettings<1-N> [<mtu>],[<socksnd>],
                    [<sockrcv>],[<tcpsnd>],
                    [<tcprcv>]]
[--natpf<1-N> [<rulename>],tcp|udp,[<hostip>],
              <hostport>,[<guestip>],<guestport>]
[--natpf<1-N> delete <rulename>]
[--nattftpprefix<1-N> <prefix>]
[--nattftpfile<1-N> <file>]
[--nattftpserver<1-N> <ip>]
[--natbindip<1-N> <ip>
[--natdnspassdomain<1-N> on|off]
[--natdnsproxy<1-N> on|off]
[--natdnshostresolver<1-N> on|off]
[--nataliasmode<1-N> default|[log],[proxyonly],
                             [sameports]]
[--macaddress<1-N> auto|<mac>]
[--mouse ps2|usb|usbtablet|usbmultitouch]
[--keyboard ps2|usb
[--uart<1-N> off|<I/O base> <IRQ>]
[--uartmode<1-N> disconnected|
                 server <pipe>|
                 client <pipe>|
                 tcpserver <port>|
                 tcpclient <hostname:port>|
                 file <file>|
                 <devicename>]
[--lpt<1-N> off|<I/O base> <IRQ>]
[--lptmode<1-N> <devicename>]
[--guestmemoryballoon <balloonsize in MB>]
[--audio none|null|dsound|oss|alsa|pulse|
         oss|pulse|coreaudio]
[--audiocontroller ac97|hda|sb16]
[--audiocodec stac9700|ad1980|stac9221|sb16]
[--clipboard disabled|hosttoguest|guesttohost|
             bidirectional]
[--draganddrop disabled|hosttoguest]
[--vrde on|off]
[--vrdeextpack default|<name>
[--vrdeproperty <name=[value]>]
[--vrdeport <hostport>]
[--vrdeaddress <hostip>]
[--vrdeauthtype null|external|guest]
[--vrdeauthlibrary default|<name>
[--vrdemulticon on|off]
[--vrdereusecon on|off]
[--vrdevideochannel on|off]
[--vrdevideochannelquality <percent>]
[--usb on|off]
[--usbehci on|off]
[--usbxhci on|off]
[--usbrename <oldname> <newname>]
[--snapshotfolder default|<path>]
[--teleporter on|off]
[--teleporterport <port>]
```

```
                        [--teleporteraddress <address|empty>
                        [--teleporterpassword <password>]
                        [--teleporterpasswordfile <file>|stdin]
                        [--tracing-enabled on|off]
                        [--tracing-config <config-string>]
                        [--tracing-allow-vm-access on|off]
                        [--usbcardreader on|off]
                        [--autostart-enabled on|off]
                        [--autostart-delay <seconds>]
                        [--videocap on|off]
                        [--videocapscreens all|<screen ID> [<screen ID> ...]]
                        [--videocapfile <filename>]
                        [--videocapres <width> <height>]
                        [--videocaprate <rate>]
                        [--videocapfps <fps>]
                        [--videocapmaxtime <ms>]
                        [--videocapmaxsize <MB>]
                        [--videocapopts <key=value> [,<key=value> ...]]
                        [--defaultfrontend default|<name>]

clonevm                 <uuid|vmname>
                        [--snapshot <uuid>|<name>]
                        [--mode machine|machineandchildren|all]
                        [--options link|keepallmacs|keepnatmacs|
                                   keepdisknames]
                        [--name <name>]
                        [--groups <group>, ...]
                        [--basefolder <basefolder>]
                        [--uuid <uuid>]
                        [--register]

import                  <ovfname/ovaname>
                        [--dry-run|-n]
                        [--options keepallmacs|keepnatmacs|importtovdi]
                        [more options]
                        (run with -n to have options displayed
                         for a particular OVF)

export                  <machines> --output|-o <name>.<ovf/ova>
                        [--legacy09|--ovf09|--ovf10|--ovf20]
                        [--manifest]
                        [--iso]
                        [--options manifest|iso|nomacs|nomacsbutnat]
                        [--vsys <number of virtual system>]
                                [--product <product name>]
                                [--producturl <product url>]
                                [--vendor <vendor name>]
                                [--vendorurl <vendor url>]
                                [--version <version info>]
                                [--description <description info>]
                                [--eula <license text>]
                                [--eulafile <filename>]

startvm                 <uuid|vmname>...
                        [--type gui|sdl|headless|separate]

controlvm               <uuid|vmname>
                        pause|resume|reset|poweroff|savestate|
                        acpipowerbutton|acpisleepbutton|
                        keyboardputscancode <hex> [<hex> ...]|
                        setlinkstate<1-N> on|off |
                        nic<1-N> null|nat|bridged|intnet|hostonly|generic|
                                 natnetwork [<devicename>] |
                        nictrace<1-N> on|off |
                        nictracefile<1-N> <filename> |
                        nicproperty<1-N> name=[value] |
                        nicpromisc<1-N> deny|allow-vms|allow-all |
```

```
                    natpf<1-N> [<rulename>],tcp|udp,[<hostip>],
                               <hostport>,[<guestip>],<guestport> |
                    natpf<1-N> delete <rulename> |
                    guestmemoryballoon <balloonsize in MB> |
                    usbattach <uuid>|<address>
                              [--capturefile <filename>] |
                    usbdetach <uuid>|<address> |
                    clipboard disabled|hosttoguest|guesttohost|
                              bidirectional |
                    draganddrop disabled|hosttoguest |
                    vrde on|off |
                    vrdeport <port> |
                    vrdeproperty <name=[value]> |
                    vrdevideochannelquality <percent> |
                    setvideomodehint <xres> <yres> <bpp>
                                     [[<display>] [<enabled:yes|no> |
                                       [<xorigin> <yorigin>]]] |
                    screenshotpng <file> [display] |
                    videocap on|off |
                    videocapscreens all|none|<screen>,[<screen>...] |
                    videocapfile <file>
                    videocapres <width>x<height>
                    videocaprate <rate>
                    videocapfps <fps>
                    videocapmaxtime <ms>
                    videocapmaxsize <MB>
                    setcredentials <username>
                                   --passwordfile <file> | <password>
                                   <domain>
                                   [--allowlocallogon <yes|no>] |
                    teleport --host <name> --port <port>
                             [--maxdowntime <msec>]
                             [--passwordfile <file> |
                              --password <password>] |
                    plugcpu <id> |
                    unplugcpu <id> |
                    cpuexecutioncap <1-100>
                    webcam <attach [path [settings]]> | <detach [path]> | <list>
                    addencpassword <id>
                                   <password file>|-
                                   [--removeonsuspend <yes|no>]
                    removeencpassword <id>
                    removeallencpasswords

discardstate        <uuid|vmname>

adoptstate          <uuid|vmname> <state_file>

snapshot            <uuid|vmname>
                    take <name> [--description <desc>] [--live]
                         [--uniquename Number,Timestamp,Space,Force] |
                    delete <uuid|snapname> |
                    restore <uuid|snapname> |
                    restorecurrent |
                    edit <uuid|snapname>|--current
                         [--name <name>]
                         [--description <desc>] |
                    list [--details|--machinereadable]
                    showvminfo <uuid|snapname>

closemedium         [disk|dvd|floppy] <uuid|filename>
                    [--delete]

storageattach       <uuid|vmname>
                    --storagectl <name>
                    [--port <number>]
                    [--device <number>]
```

```
                        [--type dvddrive|hdd|fdd]
                        [--medium none|emptydrive|additions|
                                  <uuid|filename>|host:<drive>|iscsi]
                        [--mtype normal|writethrough|immutable|shareable|
                                 readonly|multiattach]
                        [--comment <text>]
                        [--setuuid <uuid>]
                        [--setparentuuid <uuid>]
                        [--passthrough on|off]
                        [--tempeject on|off]
                        [--nonrotational on|off]
                        [--discard on|off]
                        [--hotpluggable on|off]
                        [--bandwidthgroup <name>]
                        [--forceunmount]
                        [--server <name>|<ip>]
                        [--target <target>]
                        [--tport <port>]
                        [--lun <lun>]
                        [--encodedlun <lun>]
                        [--username <username>]
                        [--password <password>]
                        [--initiator <initiator>]
                        [--intnet]

storagectl              <uuid|vmname>
                        --name <name>
                        [--add ide|sata|scsi|floppy|sas|usb|pcie]
                        [--controller LSILogic|LSILogicSAS|BusLogic|
                                      IntelAHCI|PIIX3|PIIX4|ICH6|I82078|
                        [             USB|NVMe]
                        [--portcount <1-n>]
                        [--hostiocache on|off]
                        [--bootable on|off]
                        [--rename <name>]
                        [--remove]

bandwidthctl            <uuid|vmname>
                        add <name> --type disk|network
                            --limit <megabytes per second>[k|m|g|K|M|G] |
                        set <name>
                            --limit <megabytes per second>[k|m|g|K|M|G] |
                        remove <name> |
                        list [--machinereadable]
                        (limit units: k=kilobit, m=megabit, g=gigabit,
                                      K=kilobyte, M=megabyte, G=gigabyte)

showmediuminfo          [disk|dvd|floppy] <uuid|filename>

createmedium            [disk|dvd|floppy] --filename <filename>
                        [--size <megabytes>|--sizebyte <bytes>]
                        [--diffparent <uuid>|<filename>
                        [--format VDI|VMDK|VHD] (default: VDI)
                        [--variant Standard,Fixed,Split2G,Stream,ESX]

modifymedium            [disk|dvd|floppy] <uuid|filename>
                        [--type normal|writethrough|immutable|shareable|
                                readonly|multiattach]
                        [--autoreset on|off]
                        [--property <name=[value]>]
                        [--compact]
                        [--resize <megabytes>|--resizebyte <bytes>]
                        [--move <full path to a new location>]
clonemedium             [disk|dvd|floppy] <uuid|inputfile> <uuid|outputfile>
                        [--format VDI|VMDK|VHD|RAW|<other>]
                        [--variant Standard,Fixed,Split2G,Stream,ESX]
                        [--existing]
```

```
mediumproperty          [disk|dvd|floppy] set <uuid|filename>
                        <property> <value>

                        [disk|dvd|floppy] get <uuid|filename>
                        <property>

                        [disk|dvd|floppy] delete <uuid|filename>
                        <property>

encryptmedium           <uuid|filename>
                        [--newpassword <file>|-]
                        [--oldpassword <file>|-]
                        [--cipher <cipher identifier>]
                        [--newpasswordid <password identifier>]

checkmediumpwd          <uuid|filename>
                        <pwd file>|-

convertfromraw          <filename> <outputfile>
                        [--format VDI|VMDK|VHD]
                        [--variant Standard,Fixed,Split2G,Stream,ESX]
                        [--uuid <uuid>]
convertfromraw          stdin <outputfile> <bytes>
                        [--format VDI|VMDK|VHD]
                        [--variant Standard,Fixed,Split2G,Stream,ESX]
                        [--uuid <uuid>]

getextradata            global|<uuid|vmname>
                        <key>|enumerate

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

setproperty             machinefolder default|<folder> |
                        hwvirtexclusive on|off |
                        vrdeauthlibrary default|<library> |
                        websrvauthlibrary default|null|<library> |
                        vrdeextpack null|<library> |
                        autostartdbpath null|<folder> |
                        loghistorycount <value>
                        defaultfrontend default|<name>
                        logginglevel <log setting>

usbfilter               add <index,0-N>
                        --target <uuid|vmname>|global
                        --name <string>
                        --action ignore|hold (global filters only)
                        [--active yes|no] (yes)
                        [--vendorid <XXXX>] (null)
                        [--productid <XXXX>] (null)
                        [--revision <IIFF>] (null)
                        [--manufacturer <string>] (null)
                        [--product <string>] (null)
                        [--remote yes|no] (null, VM filters only)
                        [--serialnumber <string>] (null)
                        [--maskedinterfaces <XXXXXXXX>]

usbfilter               modify <index,0-N>
                        --target <uuid|vmname>|global
                        [--name <string>]
                        [--action ignore|hold] (global filters only)
                        [--active yes|no]
                        [--vendorid <XXXX>|""]
                        [--productid <XXXX>|""]
                        [--revision <IIFF>|""]
```

```
                        [--manufacturer <string>|""]
                        [--product <string>|""]
                        [--remote yes|no] (null, VM filters only)
                        [--serialnumber <string>|""]
                        [--maskedinterfaces <XXXXXXXX>]

usbfilter               remove <index,0-N>
                        --target <uuid|vmname>|global

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

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

guestproperty           get <uuid|vmname>
                        <property> [--verbose]

guestproperty           set <uuid|vmname>
                        <property> [<value> [--flags <flags>]]

guestproperty           delete|unset <uuid|vmname>
                        <property>

guestproperty           enumerate <uuid|vmname>
                        [--patterns <patterns>]

guestproperty           wait <uuid|vmname> <patterns>
                        [--timeout <msec>] [--fail-on-timeout]

guestcontrol            <uuid|vmname> [--verbose|-v] [--quiet|-q]
                          [--username <name>] [--domain <domain>]
                          [--passwordfile <file> | --password <password>]

                          run [common-options]
                          [--exe <path to executable>] [--timeout <msec>]
                          [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
                          [--ignore-operhaned-processes] [--profile]
                          [--no-wait-stdout|--wait-stdout]
                          [--no-wait-stderr|--wait-stderr]
                          [--dos2unix] [--unix2dos]
                          -- <program/arg0> [argument1] ... [argumentN]]

                          start [common-options]
                          [--exe <path to executable>] [--timeout <msec>]
                          [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
                          [--ignore-operhaned-processes] [--profile]
                          -- <program/arg0> [argument1] ... [argumentN]]

                          copyfrom [common-options]
                          [--dryrun] [--follow] [-R|--recursive]
                          <guest-src0> [guest-src1 [...]] <host-dst>

                          copyfrom [common-options]
                          [--dryrun] [--follow] [-R|--recursive]
                          [--target-directory <host-dst-dir>]
                          <guest-src0> [guest-src1 [...]]

                          copyto [common-options]
                          [--dryrun] [--follow] [-R|--recursive]
                          <host-src0> [host-src1 [...]] <guest-dst>

                          copyto [common-options]
                          [--dryrun] [--follow] [-R|--recursive]
                          [--target-directory <guest-dst>]
                          <host-src0> [host-src1 [...]]
```

```
                          mkdir|createdir[ectory] [common-options]
                          [--parents] [--mode <mode>]
                          <guest directory> [...]

                          rmdir|removedir[ectory] [common-options]
                          [-R|--recursive]
                          <guest directory> [...]

                          removefile|rm [common-options] [-f|--force]
                          <guest file> [...]

                          mv|move|ren[ame] [common-options]
                          <source> [source1 [...]] <dest>

                          mktemp|createtemp[orary] [common-options]
                          [--secure] [--mode <mode>] [--tmpdir <directory>]
                          <template>

                          stat [common-options]
                          <file> [...]

guestcontrol             <uuid|vmname> [--verbose|-v] [--quiet|-q]

                          list <all|sessions|processes|files> [common-opts]

                          closeprocess [common-options]
                          <   --session-id <ID>
                            | --session-name <name or pattern>
                          <PID1> [PID1 [...]]

                          closesession [common-options]
                          <  --all | --session-id <ID>
                            | --session-name <name or pattern> >

                          updatega|updateguestadditions|updateadditions
                          [--source <guest additions .ISO>]
                          [--wait-start] [common-options]
                          [-- [<argument1>] ... [<argumentN>]]

                          watch [common-options]

metrics                  list [*|host|<vmname> [<metric_list>]]
                                              (comma-separated)

metrics                  setup
                         [--period <seconds>] (default: 1)
                         [--samples <count>] (default: 1)
                         [--list]
                         [*|host|<vmname> [<metric_list>]]

metrics                  query [*|host|<vmname> [<metric_list>]]

metrics                  enable
                         [--list]
                         [*|host|<vmname> [<metric_list>]]

metrics                  disable
                         [--list]
                         [*|host|<vmname> [<metric_list>]]

metrics                  collect
                         [--period <seconds>] (default: 1)
                         [--samples <count>] (default: 1)
                         [--list]
                         [--detach]
                         [*|host|<vmname> [<metric_list>]]
```

```
natnetwork                add --netname <name>
                          --network <network>
                          [--enable|--disable]
                          [--dhcp on|off]
                          [--port-forward-4 <rule>]
                          [--loopback-4 <rule>]
                          [--ipv6 on|off]
                          [--port-forward-6 <rule>]
                          [--loopback-6 <rule>]

natnetwork                remove --netname <name>

natnetwork                modify --netname <name>
                          [--network <network>]
                          [--enable|--disable]
                          [--dhcp on|off]
                          [--port-forward-4 <rule>]
                          [--loopback-4 <rule>]
                          [--ipv6 on|off]
                          [--port-forward-6 <rule>]
                          [--loopback-6 <rule>]

natnetwork                start --netname <name>

natnetwork                stop --netname <name>

natnetwork                list [<pattern>]

hostonlyif                ipconfig <name>
                          [--dhcp |
                          --ip<ipv4> [--netmask<ipv4> (def: 255.255.255.0)] |
                          --ipv6<ipv6> [--netmasklengthv6<length> (def: 64)]]
                          create |
                          remove <name>

dhcpserver                add|modify --netname <network_name> |
                                     --ifname <hostonly_if_name>
                          [--ip <ip_address>
                          --netmask <network_mask>
                          --lowerip <lower_ip>
                          --upperip <upper_ip>]
                          [--enable | --disable]

dhcpserver                remove --netname <network_name> |
                                 --ifname <hostonly_if_name>

usbdevsource              add <source name>
                          --backend <backend>
                          --address <address>
usbdevsource              remove <source name>

VBoxManage debugvm <uuid|vmname> dumpvmcore [--filename=name]
VBoxManage debugvm <uuid|vmname> info <item> [args...]
VBoxManage debugvm <uuid|vmname> injectnmi
VBoxManage debugvm <uuid|vmname> log [[--release] | [--debug]] [group-settings...]
VBoxManage debugvm <uuid|vmname> logdest [[--release] | [--debug]] [destinations...]
VBoxManage debugvm <uuid|vmname> logflags [[--release] | [--debug]] [flags...]
VBoxManage debugvm <uuid|vmname> osdetect
VBoxManage debugvm <uuid|vmname> osinfo
VBoxManage debugvm <uuid|vmname> osdmesg [--lines=lines]
VBoxManage debugvm <uuid|vmname> getregisters [--cpu=id] [reg-set.reg-name...]
VBoxManage debugvm <uuid|vmname> setregisters [--cpu=id] [reg-set.reg-name=value...]
VBoxManage debugvm <uuid|vmname> show [[--human-readable] | [--sh-export] | [--sh-eval] | [--cmd-set]]
    [settings-item...]
VBoxManage debugvm <uuid|vmname> stack [--cpu=id]
VBoxManage debugvm <uuid|vmname> statistics [--reset] [--descriptions] [--pattern=pattern]
```

```
VBoxManage extpack install [--replace] <tarball>
VBoxManage extpack uninstall [--force] <name>
VBoxManage extpack cleanup
```

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.

8.3 General options

- `-v|--version`: show the version of this tool and exit.
- `--nologo`: suppress the output of the logo information (useful for scripts)
- `--settingspw`: specifiy a settings password
- `--settingspwfile`: specify a file containing the settings password.

The settings password is used for certain settings which need to be stored encrypted for security reasons. At the moment, the only encrypted setting is the iSCSI initiator secret (see chapter 8.18, *VBoxManage storageattach*, page 150 for details). As long as no settings password is specified, this information is stored in **plain text**. After using the `--settingspw|--settingspwfile` option once, it must be always used, otherwise the encrypted setting cannot be unencrypted.

8.4 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` lists all virtual machines currently registered with VirtualBox. By default this displays a compact list with each VM's name and UUID; if you also specify `--long` or `-l`, this will be a detailed list as with the `showvminfo` command (see below).
- `runningvms` lists all currently running virtual machines by their unique identifiers (UUIDs) in the same format as with `vms`.
- `ostypes` lists all guest operating systems presently known to VirtualBox, along with the identifiers used to refer to them with the `modifyvm` command.
- `hostdvds`, `hostfloppies`, respectively, list DVD, floppy, bridged networking and host-only networking interfaces on the host, along with the name used to access them from within VirtualBox.
- `intnets` displays information about the internal networks.
- `bridgedifs`, `hostonlyifs`, `natnets` and `dhcpservers`, respectively, list bridged network interfaces, host-only network interfaces, NAT network interfaces and DHCP servers currently available on the host. Please see chapter 6, *Virtual networking*, page 92 for details on these.
- `hostinfo` displays information about the host system, such as CPUs, memory size and operating system version.
- `hostcpuids` dumps the CPUID parameters for the host CPUs. This can be used for a more fine grained analyis of the host's virtualization capabilities.

- `hddbackends` lists all known virtual disk back-ends of VirtualBox. For each such format (such as VDI, VMDK or RAW), this lists the back-end's capabilities and configuration.
- `hdds`, `dvds` and `floppies` all give you information about virtual disk images currently in use by VirtualBox, including all their settings, the unique identifiers (UUIDs) associated with them by VirtualBox and all files associated with them. This is the command-line equivalent of the Virtual Media Manager; see chapter 5.3, *The Virtual Media Manager*, page 83.
- `usbhost` 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.
- `extpacks` displays all VirtualBox extension packs currently installed; see chapter 1.5, *Installing VirtualBox and extension packs*, page 16 and chapter 8.41, *VBoxManage extpack*, page 183 for more information.
- `groups` displays details of the VM Groups; see chapter 1.9, *Using VM groups*, page 26 for more information.
- `webcams` displays a list of webcams attached to the running VM. The output format is a list of absolute paths or aliases that were used for attaching the webcams to the VM using the 'webcam attach' command.
- `screenshotformats` displays a list of available screenshot formats.

8.5 VBoxManage showvminfo

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

You will get information that resembles the following example.

```
$ VBoxManage showvminfo "Windows XP"
VirtualBox Command Line Management Interface Version 5.1.22
(C) 2005-2017 Oracle Corporation
All rights reserved.

Name:            Windows XP
Guest OS:        Other/Unknown
UUID:            1bf3464d-57c6-4d49-92a9-a5cc3816b7e7
Config file:     /home/username/.config/VirtualBox/Machines/Windows XP/Windows XP.xml
Memory size:     512MB
VRAM size:       12MB
Number of CPUs:  2
Boot menu mode:  message and menu
Boot Device (1): DVD
Boot Device (2): HardDisk
Boot Device (3): Not Assigned
Boot Device (4): Not Assigned
ACPI:            on
IOAPIC:          on
...
```

Use the `--machinereadable` option to produce the same output, but in machine readable format: property="value" on a line by line basis, e.g.:

```
name="VBoxSDL --startvm OL7.2"
groups="/"
ostype="Oracle (64-bit)"
UUID="457af700-bc0a-4258-aa3c-13b03da171f2"
...
```

8.6 VBoxManage registervm / unregistervm

The `registervm` command allows you to import a virtual machine definition in an XML file into VirtualBox. 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, the following files will automatically be deleted as well:

1. all hard disk image files, including differencing files, which are used by the machine and not shared with other machines;
2. saved state files that the machine created, if any (one if the machine was in "saved" state and one for each online snapshot);
3. the machine XML file and its backups;
4. the machine log files, if any;
5. the machine directory, if it is empty after having deleted all the above.

8.7 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 `.config/VirtualBox/Machines` folder - this folder name may vary depending on the operating system and the version of VirtualBox which you are using), 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>` option is used, 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.

If the `--group <group>, ...` option is used, the machine will be assigned membership of the specified VM groups in the list. Note that group ids always start with a `/` and can be nested. By default, VMs are always assigned membership of the group `/`.

If the `--ostype <ostype>`: option is used, <ostype> specifies the guest operating system to run in the VM. To learn about the available OS options, run `VBoxManage list ostypes` .

If the `--uuid <uuid>`: option is used, <uuid> specifies the VM uuid. This must be unique within the namespace of the host, or that of the VM Group if it is assigned to a VM group membership. By default, a unique uuid within the appropriate namespace is automatically generated.

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.

8.8 VBoxManage modifyvm

This command changes the properties of a registered virtual machine which is not running. Most of the properties that this command makes available correspond to the VM settings that VirtualBox graphical user interface displays in each VM's "Settings" dialog; these were described in chapter 3, *Configuring virtual machines*, page 45. Some of the more advanced settings, however, are only available through the `VBoxManage` interface.

These commands require that the machine is powered off (neither running nor in "saved" state). Some machine settings can also be changed while a machine is running; those settings will then have a corresponding subcommand with the `VBoxManage controlvm` subcommand (see chapter 8.13, *VBoxManage controlvm*, page 143).

8.8.1 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.
- `--groups <group>, ...`: This changes the group membership of a VM. Groups always start with a `/` and can be nested. By default VMs are in group `/`.
- `--description <desc>`: This changes the VM's description, which is a way to record details about the VM in a way which is meaningful for the user. The GUI interprets HTML formatting, the command line allows arbitrary strings potentially containing multiple lines.
- `--ostype <ostype>`: This specifies what guest operating system is supposed to run in the VM. To learn about the various identifiers that can be used here, use `VBoxManage list ostypes`.
- `--iconfile <filename>`: This specifies the absolute path on the host file system for the VirtualBox icon to be displayed in the VM.
- `--memory <memorysize>`: This sets the amount of RAM, in MB, that the virtual machine should allocate for itself from the host. See the remarks in chapter 1.7, *Creating your first virtual machine*, page 18 for more information.
- `--pagefusion on|off`: Enables/disables (default) the Page Fusion feature. The Page Fusion feature minimises memory duplication between VMs with similar configurations running on the same host. See chapter 4.9.2, *Page Fusion*, page 77 for details.
- `--vram <vramsize>`: This sets the amount of RAM that the virtual graphics card should have. See chapter 3.5, *Display settings*, page 51 for details.
- `--acpi on|off`; `--ioapic on|off`: These two determine whether the VM should have ACPI and I/O APIC support, respectively; see chapter 3.4.1, *"Motherboard" tab*, page 49 for details.

- `--pciattach <host PCI address [@ guest PCI bus address]>`: Attaches a specified PCI network controller on the host to a PCI bus (can specify) on the guest. See chapter 9.6, *PCI passthrough*, page 194 for details.
- `--pcidetach <host PCI address>`: Detaches a specified PCI network controller on the host from the attached PCI bus on the guest. See chapter 9.6, *PCI passthrough*, page 194 for details.
- `--hardwareuuid <uuid>`: The UUID presented to the guest via memory tables (DMI/SMBIOS), hardware and guest properties. By default this is the same as the VM uuid. Useful when cloning a VM. Teleporting takes care of this automatically.
- `--cpus <cpucount>`: This sets the number of virtual CPUs for the virtual machine (see chapter 3.4.2, *"Processor" tab*, page 51). If CPU hot-plugging is enabled (see below), this then sets the *maximum* number of virtual CPUs that can be plugged into the virtual machines.
- `--cpuhotplug on|off`: This enables CPU hot-plugging. When enabled, virtual CPUs can be added to and removed from a virtual machine while it is running. See chapter 9.5, *CPU hot-plugging*, page 193 for more information.
- `--plugcpu|unplugcpu <id>`: If CPU hot-plugging is enabled (see above), this adds a virtual CPU to the virtual machines (or removes one). `<id>` specifies the index of the virtual CPU to be added or removed and must be a number from 0 to the maximum no. of CPUs configured with the `--cpus` option. CPU 0 can never be removed.
- `--cpuexecutioncap <1-100>`: This setting controls how much cpu time a virtual CPU can use. A value of 50 implies a single virtual CPU can use up to 50% of a single host CPU.
- `--pae on|off`: This enables/disables PAE (see chapter 3.4.2, *"Processor" tab*, page 51).
- `--longmode on|off`: This enables/disables long mode (see chapter 3.4.2, *"Processor" tab*, page 51).
- `--cpu-profile <host|intel 80[86|286|386]>`: Indicate the use of a profile for guest cpu emulation. Specify either one based on the host system CPU (host), or one from a number of older Intel Micro-architectures - 8086, 80286, 80386.
- `--hpet on|off`: This enables/disables a High Precision Event Timer (HPET) which can replace the legacy system timers. This is turned off by default. Note that Windows supports a HPET only from Vista onwards.
- `--hwvirtex on|off`: This enables or disables the use of hardware virtualization extensions (Intel VT-x or AMD-V) in the processor of your host system; see chapter 10.3, *Hardware vs. software virtualization*, page 239.
- `--triplefaultreset on|off`: This setting allows to reset the guest instead of triggering a Guru Meditation. Some guests raise a triple fault to reset the CPU so sometimes this is desired behavior. Works only for non-SMP guests.
- `--apic on|off`: This setting enables(default)/disables IO APIC. With I/O APIC, operating systems can use more than 16 interrupt requests (IRQs) thus avoiding IRQ sharing for improved reliability. See chapter 3.4.1, *"Motherboard" tab*, page 49.
- `--x2apic on|off`: This setting enables(default)/disables CPU x2APIC support. CPU x2APIC support helps operating systems run more efficiently on high core count configurations, and optimizes interrupt distribution in virtualized environments. Disable when using host/guest operating systems incompatible with x2APIC support.

- `--paravirtprovider none|default|legacy|minimal|hyperv|kvm`: This setting specifies which paravirtualization interface to provide to the guest operating system. Specifying `none` explicitly turns off exposing any paravirtualization interface. The option `default`, will pick an appropriate interface depending on the guest OS type while starting the VM. This is the default option chosen while creating new VMs. The `legacy` option is chosen for VMs which were created with older VirtualBox versions and will pick a paravirtualization interface while starting the VM with VirtualBox 5.0 and newer. The `minimal` provider is mandatory for Mac OS X guests, while `kvm` and `hyperv` are recommended for Linux and Windows guests respectively. These options are explained in detail under chapter 10.4, *Paravirtualization providers*, page 240.

- `--paravirtdebug <key=value> [,<key=value> ...]`: This setting specifies debugging options specific to the paravirtualization provider configured for this VM. Please refer to the provider specific options under chapter 9.32, *Paravirtualized debugging*, page 228 for a list of supported key-value pairs for each provider.

- `--nestedpaging on|off`: If hardware virtualization is enabled, this additional setting enables or disables the use of the nested paging feature in the processor of your host system; see chapter 10.3, *Hardware vs. software virtualization*, page 239.

- `--largepages on|off`: If hardware virtualization *and* nested paging are enabled, for Intel VT-x only, an additional performance improvement of up to 5% can be obtained by enabling this setting. This causes the hypervisor to use large pages to reduce TLB use and overhead.

- `--vtxvpid on|off`: If hardware virtualization is enabled, for Intel VT-x only, this additional setting enables or disables the use of the tagged TLB (VPID) feature in the processor of your host system; see chapter 10.3, *Hardware vs. software virtualization*, page 239.

- `--vtxux on|off`: If hardware virtualization is enabled, for Intel VT-x only, this setting enables or disables the use of the unrestricted guest mode feature for executing your guest.

- `--accelerate3d on|off`: This enables, if the Guest Additions are installed, whether hardware 3D acceleration should be available; see chapter 4.5.1, *Hardware 3D acceleration (OpenGL and Direct3D 8/9)*, page 72.

- `--accelerate2dvideo on|off`: This enables, if the Guest Additions are installed, whether 2D video acceleration should be available; see chapter 4.5.2, *Hardware 2D video acceleration for Windows guests*, page 73.

- `--chipset piix3|ich9`: By default VirtualBox emulates an Intel PIIX3 chipset. Usually there is no reason to change the default setting unless it is required to relax some of its constraints; see chapter 3.4.1, *"Motherboard" tab*, page 49.

- 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 without color space information (Windows 3.0 format). The image must not be bigger than 640 x 480.

- `--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.

- `--biosapic x2apic|apic|disabled`: This specifies the firmware APIC level to be used. Options are: x2apic, apic or disabled (no apic or x2apic) respectively.

  Note that if x2apic is specified and x2apic is unsupported by the VCPU, biosapic downgrades to apic, if supported - otherwise to 'disabled'. Similarly, if apic is specified, and apic is unsupported results in a downgrade to 'disabled'.

- `--biossystemtimeoffset <ms>`: This specifies a fixed time offset (milliseconds) of the guest relative to the host time. If the offset is positive, the guest time runs ahead of the host time.

- `--biospxedebug on|off`: This option enables additional debugging output when using the Intel PXE boot ROM. The output will be written to the release log file (chapter 12.1.2, *Collecting debugging information*, page 247.

- `--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.

- `--rtcuseutc on|off`: This option lets the real-time clock (RTC) operate in UTC time (see chapter 3.4.1, *"Motherboard" tab*, page 49).

- `--graphicscontroller none|vboxvga|vmsvga`: This option specifies use of a graphics controller, and type chosen from vboxvga or vmsvga. chapter 3.4.1, *"Motherboard" tab*, page 49).

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

- `--firmware bios|efi|efi32|efi64`: Specifies which firmware to be used to boot the VM: Available options are BIOS, or one of the EFI options: efi, efi32 or efi64. Use EFI options with care.

- `--guestmemoryballoon <size>` sets the default size of the guest memory balloon, that is, memory allocated by the VirtualBox Guest Additions from the guest operating system and returned to the hypervisor for re-use by other virtual machines. `<size>` must be specified in megabytes. The default size is 0 megabytes. For details, see chapter 4.9.1, *Memory ballooning*, page 76.

- `--defaultfrontend default|<name>`: This allows you to specify the default frontend which will be used when starting this VM; see chapter 8.12, *VBoxManage startvm*, page 142 for details.

8.8.2 Networking settings

The following networking settings are available through `VBoxManage modifyvm`. With all these settings, the decimal number directly following the option name ("1-N" in the list below) specifies the virtual network adapter whose settings should be changed.

- `--nic<1-N> none|null|nat|natnetwork|bridged|intnet|hostonly|generic`: 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`), use the new network address translation engine (`natnetwork`), bridged networking (`bridged`) or communicate with other virtual machines using internal networking (`intnet`), host-only networking (`hostonly`), or access rarely used sub-modes (`generic`). These options correspond to the modes which are described in detail in chapter 6.2, *Introduction to networking modes*, page 93.

- `--nictype<1-N> Am79C970A|Am79C973|82540EM|82543GC|82545EM|virtio`: 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 92.

- `--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 by dumping it to a file, for debugging purposes.

  With `--nictrace<1-N> on|off`, you can enable network tracing for a particular virtual network card.

  If enabled, you must specify with `--nictracefile<1-N> <filename>` what file the trace should be logged to.

- `--nicproperty<1-N> <paramname>="paramvalue"`: This option, in combination with "nicgenericdrv" allows you to pass parameters to rarely-used network backends.

  Those parameters are backend engine-specific, and are different between UDP Tunnel and the VDE backend drivers. For example, please see chapter 6.8, *UDP Tunnel networking*, page 100.

- `--nicspeed<1-N> <kbps>`: If generic networking has been enabled for a virtual network card (see the `--nic` option above; otherwise this setting has no effect), this mode enables access to rarely used networking sub-modes, such as VDE network or UDP Tunnel. This option specifies the throughput rate in KBytes/sec.

- `--nicbootprio<1-N> <priority>`: This specifies the order in which NICs are tried for booting over the network (using PXE). The priority is an integer in the 0 to 4 range. Priority 1 is the highest, priority 4 is low. Priority 0, which is the default unless otherwise specified, is the lowest.

  Note that this option only has effect when the Intel PXE boot ROM is used.

- `--nicpromisc<1-N> deny|allow-vms|allow-all`: This allows you, for each of the VM's virtual network cards, to specify how the promiscuous mode is handled. This setting is only relevant for bridged networking. `deny` (default setting) hides any traffic not intended for this VM. `allow-vms` hides all host traffic from this VM but allows the VM to see traffic from/to other VMs. `allow-all` removes this restriction completely.

- `--nicbandwidthgroup<1-N> none|<name>`: This removes/adds an assignment of a bandwidth group from/to the specified virtual network interface. Specifying `none` removes any current bandwidth group assignment from the specified virtual network interface. Specifying `<name>` adds an assignment of a bandwidth group to the specified virtual network interface.

  For details, please see chapter 6.10, *Limiting bandwidth for network I/O*, page 101.

- `--bridgeadapter<1-N> none|<devicename>`: If bridged 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 details, please see chapter 6.5, *Bridged networking*, page 97.

- `--hostonlyadapter<1-N> none|<devicename>`: If host-only 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-only networking interface the given virtual network interface will use. For details, please see chapter 6.7, *Host-only networking*, page 99.

- `--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 6.6, *Internal networking*, page 98).
- `--nat-network<1-N> <network name>`: If the networking type is set to `natnetwork` (not `nat`) then this setting specifies the name of the NAT network this adapter is connected to.
- `--nicgenericdrv<1-N> <backend driver>`: If generic networking has been enabled for a virtual network card (see the `--nic` option above; otherwise this setting has no effect), this mode allows you to access rarely used networking sub-modes, such as VDE network or UDP Tunnel.
- `--macaddress<1-N> auto|<mac>`: With this option you can set the MAC address of a particular network adapter on the VM. Normally, each network adapter is assigned a random address by VirtualBox at VM creation.

8.8.2.1 NAT Networking settings.

The following NAT networking settings are available through `VBoxManage modifyvm`. With all these settings, the decimal number directly following the option name ("1-N" in the list below) specifies the virtual network adapter whose settings should be changed.

- `--natnet<1-N> <network>|default`: If the networking type is set to `nat` (not `natnetwork`) then this setting specifies the IP address range to be used for this network. See chapter 9.11, *Fine-tuning the VirtualBox NAT engine*, page 201 for an example.
- `--natpf<1-N> [<name>],tcp|udp,[<hostip>],<hostport>,[<guestip>], <guestport>`: This option defines a NAT port-forwarding rule (please see chapter 6.3.1, *Configuring port forwarding with NAT*, page 94 for details).
- `--natpf<1-N> delete <name>`: This option deletes a NAT port-forwarding rule (please see chapter 6.3.1, *Configuring port forwarding with NAT*, page 94 for details).
- `--nattftpprefix<1-N> <prefix>`: This option defines a prefix for the built-in TFTP server, i.e. where the boot file is located (please see chapter 6.3.2, *PXE booting with NAT*, page 95 and chapter 9.11.2, *Configuring the boot server (next server) of a NAT network interface*, page 202 for details).
- `--nattftpfile<1-N> <bootfile>`: This option defines the TFT boot file (please see chapter 9.11.2, *Configuring the boot server (next server) of a NAT network interface*, page 202 for details).
- `--nattftpserver<1-N> <tftpserver>`: This option defines the TFTP server address to boot from (please see chapter 9.11.2, *Configuring the boot server (next server) of a NAT network interface*, page 202 for details).
- `--nattbindip<1-N> <ip;>`: VirtualBox's NAT engine normally routes TCP/IP packets through the default interface assigned by the host's TCP/IP stack. Use this option to instruct the NAT engine to bind to a specified IP address instead. (Please see chapter 9.11.3, *Tuning TCP/IP buffers for NAT*, page 202 for details).
- `--natdnspassdomain<1-N> on|off`: This option specifies whether the built-in DHCP server passes the domain name for network name resolution.
- `--natdnsproxy<1-N> on|off`: This option makes the NAT engine proxy all guest DNS requests to the host's DNS servers (please see chapter 9.11.5, *Enabling DNS proxy in NAT mode*, page 202 for details).

- `--natdnshostresolver<1-N> on|off`: This option makes the NAT engine use the host's resolver mechanisms to handle DNS requests (please see chapter 9.11.5, *Enabling DNS proxy in NAT mode*, page 202 for details).
- `--natsettings<1-N> [<mtu>],[<socksnd>],[<sockrcv>],[<tcpsnd>], [<tcprcv>]`: This option controls several NAT settings (please see chapter 9.11.3, *Tuning TCP/IP buffers for NAT*, page 202 for details).
- `--nataliasmode<1-N> default|[log],[proxyonly],[sameports]`: This option defines behaviour of NAT engine core: log - enables logging, proxyonly - switches of aliasing mode makes NAT transparent, sameports enforces NAT engine to send packets via the same port as they originated on, default - disable all mentioned modes above . (please see chapter 9.11.7, *Configuring aliasing of the NAT engine*, page 203 for details).

8.8.3 Miscellaneous settings

The following other hardware settings, such as serial port, audio, clipboard, drag and drop, monitor and USB settings are available through `VBoxManage modifyvm`:

- `--mouse <ps2|usb|usbtablet|usbmultitouch>`: Specifies the mode of the mouse to be used in the VM. Available options are: ps2, usb, usbtablet, usbmultitouch.
- `--keyboard <ps2|usb>`: Specifies the mode of the keyboard to be used in the VM. Available options are: ps2, usb.
- `--uart<1-N> off|<I/O base> <IRQ>`: With this option you can configure virtual serial ports for the VM; see chapter 3.9, *Serial ports*, page 55 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 3.9, *Serial ports*, page 55, 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.
  - `tcpserver <port>`: This tells VirtualBox to create a TCP socket on the host with TCP `<port>` and connect the virtual serial device to it. Note that UNIX-like systems require ports over 1024 for normal users.
  - `tcpclient <hostname:port>`: This operates just like `tcpserver ...`, except that the TCP socket is not created by VirtualBox, but assumed to exist already.
  - `file <file>`: This redirects the serial port output to a raw file <file> specified by its absolute path on the host file system.
  - `<devicename>`: If, instead of the above, 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.

- `--lptmode<1-N> <Device>`: Specifies the Device Name of the parallel port that the Parallel Port feature will be using. Use this *before* `--lpt`. This feature is host operating system specific. For Windows hosts, use a device name like *lpt1* while on Linux hosts you have to use a device name like */dev/lp0*

- `--lpt<1-N> <I/O base> <IRQ>`: Specifies the I/O address of the parallel port and the IRQ number that the Parallel Port feature will be using. Use this *after* `--lptmod`. I/O base address and IRQ are the values that guest sees i.e. the values avalable under guest Device Manager.

- `--audio none|null|dsound|oss|alsa|pulse|coreaudio`: With this setting, you can specify whether the VM should have audio support, and – if so – which type. The list of supported audio types depends on the host and can be determined with `VBoxManage modifyvm`.

- `--audiocontroller ac97|hda|sb16`: With this option, you can specify the audio controller to be used with this VM.

- `--audiocodec stac9700|ad1980|stac9221|sb16`: With this option, you can specify the audio codec to be used with this VM.

- `--clipboard disabled|hosttoguest|guesttohost|bidirectional`: With this setting, you can select if and how the guest or host operating system's clipboard should be shared with the host or guest; see chapter 3.3, *General settings*, page 48. This requires that the Guest Additions be installed in the virtual machine.

- `--draganddrop disabled|hosttoguest|guesttohost|bidirectional`: With this setting, you can select the current drag and drop mode being used between the host and the virtual machine; see chapter 4.4, *Drag and Drop*, page 70. This requires that the Guest Additions be installed in the virtual machine.

- `--monitorcount <count>`: This enables multi-monitor support; see chapter 3.5, *Display settings*, page 51.

- `--usb on|off`: This option enables or disables the VM's virtual USB controller; see chapter 3.10.1, *USB settings*, page 56 for details.

- `--usbehci on|off`: This option enables or disables the VM's virtual USB 2.0 controller; see chapter 3.10.1, *USB settings*, page 56 for details.

- `--usbxhci on|off`: This option enables or disables the VM's virtual USB 3.0 controller; see chapter 3.10.1, *USB settings*, page 56 for details.

- `--usbrename <oldname> <newname>`: This option enables renaming of the VM's virtual USB controller from <oldname> to <newname>.

8.8.4 Video Capture settings

The following settings for changing video recording parameters are available through `VBoxManage modifyvm`.

- `--videocap on|off`: This option enables or disables recording a VM session into a WebM/VP8 file. If this option is enabled, recording will start when the VM session is started.

- `--videocapscreens all|<screen ID> [<screen ID> ...]`: This option allows to specify which screens of the VM are being recorded. Each screen is recorded into a separate file.

- `--videocapfile <filename>`: This option sets the filename VirtualBox uses to save the recorded content.
- `--videocapres <width>x<height>`: This option sets the resolution (in pixels) of the recorded video.
- `--videocaprate <rate>`: This option sets the bitrate in kilobits (kb) per second. Increasing this value makes the video look better for the cost of an increased file size.
- `--videocapfps <fps>`: This option sets the maximum number of frames per second (FPS) to be recorded. Frames with a higher frequency will be skipped. Reducing this value increases the number of skipped frames and reduces the file size.
- `--videocapmaxtime <ms>`: This option sets the maximum time in milliseconds the video capturing will be enabled since activation. The capturing stops when the defined time interval has elapsed. If this value is zero the capturing is not limited by time.
- `--videocapmaxsize <MB>`: This option limits the maximum size of the captured video file (in MB). The capturing stops when the file size has reached the specified size. If this value is zero the capturing will not be limited by file size.
- `--videocapopts <key=value> [,<key=value> ...]`: This format can be used to specify additional video capturing options. These options only are for advanced users and must be specified in a comma-separated key=value format, e.g. `foo=bar,a=b`.

8.8.5 Remote machine settings

The following settings that affect remote machine behavior are available through `VBoxManage modifyvm`:

- `--vrde on|off`: This enables or disables the VirtualBox remote desktop extension (VRDE) server.
- `--vrdeproperty "TCP/Ports|Address=<value>"` sets the port number(s) and IP address on the VM that the VRDE server can bind to.
  - For TCP/Ports, <value> should be a port or a range of ports that the VRDE server can bind to; "default" or "0" means port 3389, the standard port for RDP. For details, see the description for the `--vrdeport` option in chapter 8.8.5, *Remote machine settings*, page 136.
  - For TCP/Address, <value> should be the IP address of the host network interface that the VRDE server will bind to. If specified, the server will accept connections only on the specified host network interface. For details, see the description for the `--vrdeaddress` option in chapter 8.8.5, *Remote machine settings*, page 136.
- `--vrdeproperty "VideoChannel/Enabled|Quality|DownscaleProtection=<value>"` sets the VRDP video redirection properties.
  - For VideoChannel/Enabled, <value> can be set to "1" switching the VRDP video channel on. For details, see chapter 7.1.9, *VRDP video redirection*, page 111.
  - For VideoChannel/Quality, <value> should be set between 10 and 100% inclusive, representing a JPEG compression level on the VRDE server video channel. Lower values mean lower quality but higher compression. For details, see chapter 7.1.9, *VRDP video redirection*, page 111.

  - For VideoChannel/DownscaleProtection, <value> can be set to "1" to enable the videochannel downscale protection feature. When enabled, if a video's size equals the shadow buffer size, then it is regarded as a full screen video, and is displayed; but if its size is between fullscreen and the downscale threshold - it is NOT displayed, as it could be an application window, which would be unreadable when downscaled. When the downscale protection feature is disabled, an attempt is always made to display videos.

- `--vrdeproperty "Client/DisableDisplay|DisableInput|DisableAudio|DisableUSB=1"`

  disables one of the VRDE server features: Display, Input, Audio or USB respectively. To re-enable a feature, use e.g. "Client/DisableDisplay=". For details, see chapter 7.1.10, *VRDP customization*, page 111.

- `--vrdeproperty "Client/DisableClipboard|DisableUpstreamAudio=1"`

  disables one of the VRDE server features: Clipboard or UpstreamAudio respectively. To re-enable a feature, use e.g. "Client/DisableClipboard=". For details, see chapter 7.1.10, *VRDP customization*, page 111.

- `--vrdeproperty "Client/DisableRDPDR=1"`

  disables the VRDE server feature: RDP device redirection for smart cards. To re-enable this feature, use "Client/DisableRDPR=".

- `--vrdeproperty "H3DRedirect/Enabled=1"`

  enables the VRDE server feature: 3D redirection. To re-disable this feature, use "H3DRedirect/Enabled=".

- `--vrdeproperty "Security/Method|ServerCertificate|ServerPrivateKey|CACertificate=<value>"` sets the desired security method/Path of server certificate, path of server private key, path of CA certificate, used for a connection.

  - `--vrdeproperty "Security/Method=<value>"` sets the desired security method, which is used for a connection. Valid values are:

    * `Negotiate` - both Enhanced (TLS) and Standard RDP Security connections are allowed. The security method is negotiated with the client. This is the default setting.
    * `RDP` - only Standard RDP Security is accepted.
    * `TLS` - only Enhanced RDP Security is accepted. The client must support TLS.

    For details, see chapter 7.1.6, *RDP encryption*, page 109.

  - `--vrdeproperty "Security/ServerCertificate=<value>"` where <value> is the absolute path of the server certificate. For details, see chapter 7.1.6, *RDP encryption*, page 109.
  - `--vrdeproperty "Security/ServerPrivateKey=<value>"` where <value> is the absolute path of the server private key. For details, see chapter 7.1.6, *RDP encryption*, page 109.
  - `--vrdeproperty "Security/CACertificate=<value>"` where <value> is the absolute path of the CA self signed certificate. For details, see chapter 7.1.6, *RDP encryption*, page 109.

- `--vrdeproperty "Audio/RateCorrectionMode|LogPath=<value>"` sets the Audio connection mode, or Path of the audio logfile.

  - `--vrdeproperty "Audio/RateCorrectionMode=<value>"` where <value> is the desired rate correction mode, allowed values are:

    * VRDP_AUDIO_MODE_VOID - no mode specified, use to unset any Audio mode already set.
    * VRDP_AUDIO_MODE_RC - rate correction mode.
    * VRDP_AUDIO_MODE_LPF - low pass filter mode.
    * VRDP_AUDIO_MODE_CS - client sync mode to prevent under/overflow of the client queue.
  - --vrdeproperty "Audio/LogPath=<value>" where <value> is the absolute path of the Audio log file.

- --vrdeextpack default|<name>: Allows to specify the library to use for to access the VM remotely. The default is to use the RDP code which is part of the Oracle VM VirtualBox Extension Pack.

- --vrdeport default|<ports>: A port or a range of ports the VRDE server can bind to; "default" or "0" means port 3389, the standard port for RDP. You can specify a comma-separated list of ports or ranges of ports. Use a dash between two port numbers to specify a range. The VRDE server will bind to **one** of available ports from the specified list. Only one machine can use a given port at a time. For example, the option --vrdeport 5000,5010-5012 will tell the server to bind to one of following ports: 5000, 5010, 5011 or 5012.

- --vrdeaddress <IP address>: The IP address of the host network interface the VRDE server will bind to. If specified, the server will accept connections only on the specified host network interface.

  The setting can be used to specify whether the VRDP server should accept either IPv4 or IPv6 or both connections:

  - only IPv4: --vrdeaddress "0.0.0.0"
  - only IPv6: --vrdeaddress "::"
  - both IPv6 and IPv4 (default): --vrdeaddress ""

- --vrdeauthtype null|external|guest: This allows you to choose whether and how authorization will be performed; see chapter 7.1.5, *RDP authentication*, page 108 for details.

- --vrdeauthlibrary default|<name>: This allos to set the library used for RDP authentication, see chapter 7.1.5, *RDP authentication*, page 108 for details.

- --vrdemulticon on|off: This enables multiple connections to the same VRDE server, if the server supports this feature; see chapter 7.1.7, *Multiple connections to the VRDP server*, page 110.

- --vrdereusecon on|off: This specifies the VRDE server behavior when multiple connections are disabled. When this option is enabled, the server will allow a new client to connect and will drop the existing connection. When this option is disabled (this is the default setting), a new connection will not be accepted if there is already a client connected to the server.

- --vrdevideochannel on|off: This enables video redirection, if it is supported by the VRDE server; see chapter 7.1.9, *VRDP video redirection*, page 111.

- --vrdevideochannelquality <percent>: Sets the image quality for video redirection; see chapter 7.1.9, *VRDP video redirection*, page 111.

8.8.6 Teleporting settings

With the following commands for `VBoxManage modifyvm` you can configure a machine to be a target for teleporting. See chapter 7.2, *Teleporting*, page 112 for an introduction.

- `--teleporter on|off`: With this setting you turn on or off whether a machine waits for a teleporting request to come in on the network when it is started. If "on", when the machine is started, it does not boot the virtual machine as it would normally; instead, it then waits for a teleporting request to come in on the port and address listed with the next two parameters.
- `--teleporterport <port>`, `--teleporteraddress <address>`: these must be used with –teleporter and tell the virtual machine on which port and address it should listen for a teleporting request from another virtual machine. `<port>` can be any free TCP/IP port number (e.g. 6000); `<address>` can be any IP address or hostname and specifies the TCP/IP socket to bind to. The default is "0.0.0.0", which means any address.
- `--teleporterpassword <password>`: if this optional argument is given, then the teleporting request will only succeed if the source machine specifies the same password as the one given with this command.
- `--teleporterpasswordfile <password>`: if this optional argument is given, then the teleporting request will only succeed if the source machine specifies the same password as the one specified in the file give with this command. Use `stdin` to read the password from stdin.
- `--cpuid <leaf> <eax> <ebx> <ecx> <edx>`: Advanced users can use this command before a teleporting operation to restrict the virtual CPU capabilities that VirtualBox presents to the guest operating system. This must be run on both the source and the target machines involved in the teleporting and will then modify what the guest sees when it executes the `CPUID` machine instruction. This might help with misbehaving applications that wrongly assume that certain CPU capabilities are present. The meaning of the parameters is hardware dependent; please refer to the AMD or Intel processor manuals.

8.8.7 Debugging settings

The following settings are only relevant for low-level VM debugging. Regular users will never need these settings.

- `--tracing-enabled on|off`: Enable the tracebuffer. This consumes some memory for the tracebuffer and adds extra overhead.
- `--tracing-config <config-string>`: Allows to configure tracing. In particular this defines which group of tracepoints are enabled.
- `--tracing-allow-vm-access on|off`: Enables/disables(default) VM access to the tracebuffer.

8.8.8 USB card reader settings

The following setting defines access to a USB Card Reader by the guest environment. USB card readers are typically used for accessing data on memory cards such as CompactFlash (CF), Secure Digital (SD) or MultiMediaCard (MMC).

- `--usbcardreader on|off`: Enables/disables the USB card reader interface.

8.8.9 Auto starting VMs during host system boot

These settings configure the VM autostart feature, which automatically starts the VM at host system boot-up. Note that there are pre-requisites that need to be addressed before using this feature. See chapter 9.24, *Starting virtual machines during system boot*, page 223 for more details.

- `--autostart on|off`: Enables/disables VM autostart at host system boot-up, using specified user name.
- `--autostart-delay <seconds>`: Specifies a delay (seconds) following host system boot-up, before VM autostarts.

8.9 VBoxManage clonevm

This command creates a full or linked copy of an existing virtual machine.

The `clonevm` subcommand takes at least the name of the virtual machine which should be cloned. The following additional settings can be used to further configure the clone VM operation:

- `--snapshot <uuid>|<name>`: Select a specific snapshot where the clone operation should refer to. Default is referring to the current state.
- `--mode machine|machineandchildren|all`: Selects the cloning mode of the operation. If `machine` is selected (the default), the current state of the VM without any snapshots is cloned. In the `machineandchildren` mode the snapshot provided by `--snapshot` and all child snapshots are cloned. If `all` is the selected mode all snapshots and the current state are cloned.
- `--options link|keepallmacs|keepnatmacs|keepdisknames`: Allows additional fine tuning of the clone operation. The first option defines that a linked clone should be created, which is only possible for a machine clone from a snapshot. The next two options allow to define how the MAC addresses of every virtual network card should be handled. They can either be reinitialized (the default), left unchanged (`keepallmacs`) or left unchanged when the network type is NAT (`keepnatmacs`). If you add `keepdisknames` all new disk images are called like the original ones, otherwise they are renamed.
- `--name <name>`: Select a new name for the new virtual machine. Default is "Original Name Clone".
- `--groups <group>, ...` Enables the clone to be assigned membership of the specified VM groups in the list. Note that group ids always start with a `/` and can be nested. By default, clones are always assigned membership of the group `/`.
- `--basefolder <basefolder>`: Select the folder where the new virtual machine configuration should be saved in.
- `--uuid <uuid>`: Select the UUID the new VM should have. This id has to be unique in the VirtualBox instance this clone should be registered. Default is creating a new UUID.
- `--register`: Automatically register the new clone in this VirtualBox installation. If you manually want to register the new VM later, see chapter 8.6, *VBoxManage registervm / unregistervm*, page 127 for instructions how to do so.

8.10 VBoxManage import

This command imports a virtual appliance in OVF format by copying the virtual disk images and creating virtual machines in VirtualBox. See chapter 1.14, *Importing and exporting virtual machines*, page 31 for an introduction to appliances.

The `import` subcommand takes at least the path name of an OVF file as input and expects the disk images, if needed, in the same directory as the OVF file. A lot of additional command-line options are supported to control in detail what is being imported and modify the import parameters, but the details depend on the content of the OVF file.

It is therefore recommended to first run the import subcommand with the `--dry-run` or `-n` option. This will then print a description of the appliance's contents to the screen how it would be imported into VirtualBox, together with the optional command-line options to influence the import behavior.

Use of the `--options link|keepallmacs|keepnatmacs|keepdisknames:` option enables additional fine tuning of the clone operation. The first option defines that a linked clone should be created, which is only possible for a machine clone from a snapshot. The next two options enable specification of how the MAC addresses of every virtual network card should be handled. They can either be reinitialized (the default), left unchanged (`keepallmacs`) or left unchanged when the network type is NAT (`keepnatmacs`). If you add `keepdisknames` all new disk images are assigned the same names as the originals, otherwise they are renamed.

As an example, here is the screen output with a sample appliance containing a Windows XP guest:

```
VBoxManage import WindowsXp.ovf --dry-run
Interpreting WindowsXp.ovf...
OK.
Virtual system 0:
 0: Suggested OS type: "WindowsXP"
    (change with "--vsys 0 --ostype <type>"; use "list ostypes" to list all)
 1: Suggested VM name "Windows XP Professional_1"
    (change with "--vsys 0 --vmname <name>")
 3: Number of CPUs: 1
    (change with "--vsys 0 --cpus <n>")
 4: Guest memory: 956 MB (change with "--vsys 0 --memory <MB>")
 5: Sound card (appliance expects "ensoniq1371", can change on import)
    (disable with "--vsys 0 --unit 5 --ignore")
 6: USB controller
    (disable with "--vsys 0 --unit 6 --ignore")
 7: Network adapter: orig bridged, config 2, extra type=bridged
 8: Floppy
    (disable with "--vsys 0 --unit 8 --ignore")
 9: SCSI controller, type BusLogic
    (change with "--vsys 0 --unit 9 --scsitype {BusLogic|LsiLogic}";
    disable with "--vsys 0 --unit 9 --ignore")
10: IDE controller, type PIIX4
    (disable with "--vsys 0 --unit 10 --ignore")
11: Hard disk image: source image=WindowsXp.vmdk,
      target path=/home/user/disks/WindowsXp.vmdk, controller=9;channel=0
    (change controller with "--vsys 0 --unit 11 --controller <id>";
    disable with "--vsys 0 --unit 11 --ignore")
```

As you can see, the individual configuration items are numbered, and depending on their type support different command-line options. The import subcommand can be directed to ignore many such items with a `--vsys X --unit Y --ignore` option, where X is the number of the virtual system (zero unless there are several virtual system descriptions in the appliance) and Y the item number, as printed on the screen.

In the above example, Item #1 specifies the name of the target machine in VirtualBox. Items #9 and #10 specify hard disk controllers, respectively. Item #11 describes a hard disk image; in this case, the additional `--controller` option indicates which item the disk image should be connected to, with the default coming from the OVF file.

You can combine several items for the same virtual system behind the same `--vsys` option. For example, to import a machine as described in the OVF, but without the sound card and without the USB controller, and with the disk image connected to the IDE controller instead of the SCSI controller, use this:

```
VBoxManage import WindowsXp.ovf
      --vsys 0 --unit 5 --ignore --unit 6 --ignore --unit 11 --controller 10
```

8.11 VBoxManage export

This command exports one or more virtual machines from VirtualBox into a virtual appliance in OVF format, including copying their virtual disk images to compressed VMDK. See chapter 1.14, *Importing and exporting virtual machines*, page 31 for an introduction to appliances.

The `export` command is simple to use: list the machine (or the machines) that you would like to export to the same OVF file and specify the target OVF file after an additional `--output` or `-o` option. Note that the directory of the target OVF file will also receive the exported disk images in the compressed VMDK format (regardless of the original format) and should have enough disk space left for them.

Beside a simple export of a given virtual machine, you can append several product information to the appliance file. Use `--product`, `--producturl`, `--vendor`, `--vendorurl`, `--version` and `--description` to specify this additional information. For legal reasons you may add a license text or the content of a license file by using the `--eula` and `--eulafile` option respectively. As with OVF import, you must use the `--vsys X` option to direct the previously mentioned options to the correct virtual machine.

For virtualization products which aren't fully compatible with the OVF standard 1.0 you can enable a OVF 0.9 legacy mode with the `--legacy09` option. Other options are –ovf09, –ovf10, –ovf20.

To specify options controlling the exact content of the appliance file, you can use `--options` to request the creation of a manifest file (encouraged, allows detection of corrupted appliances on import), the additional export of DVD images, and the exclusion of MAC addresses. You can specify a list of options, e.g. `--options manifest,nomacs`. For details, check the help output of `VBoxManage export`.

8.12 VBoxManage startvm

This command starts a virtual machine that is currently in the "Powered off" or "Saved" states.

The optional `--type` specifier determines whether the machine will be started in a window or whether the output should go through `VBoxHeadless`, with VRDE enabled or not; see chapter 7.1.2, *VBoxHeadless, the remote desktop server*, page 105 for more information. The list of types is subject to change, and it's not guaranteed that all types are accepted by any product variant.

The global or per-VM default value for the VM frontend type will be taken if the type is not explicitly specified. If none of these are set, the GUI variant will be started.

The following values are allowed:

`gui` Starts a VM showing a GUI window. This is the default.

`headless` Starts a VM without a window for remote display only.

`sdl` Starts a VM with a minimal GUI and limited features.

`separate` Starts a VM with detachable UI (technically it is a headless VM with user interface in a separate process). This is an experimental feature as it lacks certain functionality at the moment (e.g. 3D acceleration will not work).

Note: If you experience problems with starting virtual machines with particular frontends and there is no conclusive error information, consider starting virtual machines directly by running the respective front-end, as this can give additional error information.

8.13 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 "Machine" 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 "Machine" menu of the GUI.)

- `VBoxManage controlvm <vm> reset` has the same effect on a virtual machine as pressing the "Reset" button on a real computer: a cold reboot of the virtual machine, 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 "Machine" menu of the GUI).

- `VBoxManage controlvm <vm> poweroff` has the same effect on a virtual machine as pulling the power cable on a real computer. 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 "Machine" 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 8.12, *VBoxManage startvm*, page 142.

- `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 "Machine" 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 8.12, *VBoxManage startvm*, page 142.

- `VBoxManage controlvm <vm> acpipowerbutton` will send an ACPI shutdown signal to the VM, as if the power button on a real computer had been pressed. So long as the VM is running a fairly modern guest operating system providing ACPI support, this should trigger a proper shutdown mechanism from within the VM.

- `VBoxManage controlvm <vm> keyboardputscancode <hex> [<hex>...]` Sends commands using keycodes to the VM. Keycodes are documented in the public domain, e.g. http://www.win.tue.nl/~aeb/linux/kbd/scancodes-1.html.

- `VBoxManage controlvm "VM name" teleport --hostname <name> --port <port> [--passwordfile <file> | --password <password>]` makes the machine the source of a teleporting operation and initiates a teleport to the given target. See chapter 7.2, *Teleporting*, page 112 for an introduction. If the optional password is specified, it must match the password that was given to the `modifyvm` command for the target machine; see chapter 8.8.6, *Teleporting settings*, page 139 for details.

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

- The `setlinkstate<1-N>` operation connects or disconnects virtual network cables from their network interfaces.
- `nic<1-N> null|nat|bridged|intnet|hostonly|generic|natnetwork[<devicename>]`: 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 connected to the host (`null`), use network address translation (`nat`), bridged networking (`bridged`) or communicate with other virtual machines using internal networking (`intnet`) or host-only networking (`hostonly`) or natnetwork networking (`natnetwork`) or access to rarely used sub-modes (`generic`). These options correspond to the modes which are described in detail in chapter 6.2, *Introduction to networking modes*, page 93.
- With the "nictrace" options, you can optionally trace network traffic by dumping it to a file, for debugging purposes.

  With `nictrace<1-N> on|off`, you can enable network tracing for a particular virtual network card.

  If enabled, you must specify with `--nictracefile<1-N> <filename>` what file the trace should be logged to.
- `nicpromisc<1-N> deny|allow-vms|allow-all`: This allows you, for each of the VM's virtual network cards, to specify how the promiscious mode is handled. This setting is only relevant for bridged networking. `deny` (default setting) hides any traffic not intended for this VM. `allow-vms` hides all host traffic from this VM but allows the VM to see traffic from/to other VMs. `allow-all` removes this restriction completely.
- `nicproperty<1-N> <paramname>="paramvalue"`: This option, in combination with "nic-genericdrv" allows you to pass parameters to rarely-used network backends.

  Those parameters are backend engine-specific, and are different between UDP Tunnel and the VDE backend drivers. For example, please see chapter 6.8, *UDP Tunnel networking*, page 100.
- `natpf<1-N> [<name>],tcp|udp,[<hostip>],<hostport>,[<guestip>], <guestport>`: This option defines a NAT port-forwarding rule (please see chapter 6.3.1, *Configuring port forwarding with NAT*, page 94 for details).
- `natpf<1-N> delete <name>`: This option deletes a NAT port-forwarding rule (please see chapter 6.3.1, *Configuring port forwarding with NAT*, page 94 for details).
- The `guestmemoryballoon<balloon size in MB>` operation changes the size of the guest memory balloon, that is, memory allocated by the VirtualBox Guest Additions from the guest operating system and returned to the hypervisor for re-use by other virtual machines. This must be specified in megabytes. For details, see chapter 4.9.1, *Memory ballooning*, page 76.
- `usbattach<uuid|address> [--capturefile <filename>]`

  and `usbdetach <uuid|address> [--capturefile <filename>]` make host USB devices visible/invisible to the virtual machine on the fly, without the need for creating filters first. The USB devices can be specified by UUID (unique identifier) or by address on the host system. Use the –capturefile option to specify the absolute path of a file for writing activity logging data.

  You can use `VBoxManage list usbhost` to locate this information.

- `clipboard disabled|hosttoguest|guesttohost|bidirectional`: With this setting, you can select if and how the guest or host operating system's clipboard should be shared with the host or guest; see chapter 3.3, *General settings*, page 48. This requires that the Guest Additions be installed in the virtual machine.

- `draganddrop disabled|hosttoguest|guesttohost|bidirectional`: With this setting, you can select the current drag and drop mode being used between the host and the virtual machine; see chapter 4.4, *Drag and Drop*, page 70. This requires that the Guest Additions be installed in the virtual machine.

- `vrde on|off` lets you enable or disable the VRDE server, if it is installed.

- `vrdeport default|<ports>` changes the port or a range of ports that the VRDE server can bind to; "default" or "0" means port 3389, the standard port for RDP. For details, see the description for the `--vrdeport` option in chapter 8.8.5, *Remote machine settings*, page 136.

- `vrdeproperty "TCP/Ports|Address=<value>"` sets the port number(s) and IP address on the VM that the VRDE server can bind to.

  - For TCP/Ports, <value> should be a port or a range of ports that the VRDE server can bind to; "default" or "0" means port 3389, the standard port for RDP. For details, see the description for the `--vrdeport` option in chapter 8.8.5, *Remote machine settings*, page 136.
  - For TCP/Address, <value> should be the IP address of the host network interface that the VRDE server will bind to. If specified, the server will accept connections only on the specified host network interface. For details, see the description for the `--vrdeaddress` option in chapter 8.8.5, *Remote machine settings*, page 136.

- `vrdeproperty "VideoChannel/Enabled|Quality|DownscaleProtection=<value>"` sets the VRDP video redirection properties.

  - For VideoChannel/Enabled, <value> can be set to "1" switching the VRDP video channel on. For details, see chapter 7.1.9, *VRDP video redirection*, page 111.
  - For VideoChannel/Quality, <value> should be set between 10 and 100% inclusive, representing a JPEG compression level on the VRDE server video channel. Lower values mean lower quality but higher compression. For details, see chapter 7.1.9, *VRDP video redirection*, page 111.
  - For VideoChannel/DownscaleProtection, <value> can be set to "1" to enable the videochannel downscale protection feature. When enabled, if a video's size equals the shadow buffer size, then it is regarded as a full screen video, and is displayed; but if its size is between fullscreen and the downscale threshold - it is NOT displayed, as it could be an application window, which would be unreadable when downscaled. When the downscale protection feature is disabled, an attempt is always made to display videos.

- `vrdeproperty "Client/DisableDisplay|DisableInput|DisableAudio|DisableUSB=1"`

  disables one of the VRDE server features: Display, Input, Audio or USB respectively. To re-enable a feature, use e.g. "Client/DisableDisplay=". For details, see chapter 7.1.10, *VRDP customization*, page 111.

- `vrdeproperty "Client/DisableClipboard|DisableUpstreamAudio=1"`

  disables one of the VRDE server features: Clipboard or UpstreamAudio respectively. To re-enable a feature, use e.g. "Client/DisableClipboard=". For details, see chapter 7.1.10, *VRDP customization*, page 111.

- `vrdeproperty "Client/DisableRDPDR=1"`

  disables the VRDE server feature: RDP device redirection for smart cards. To re-enable this feature, use "Client/DisableRDPR=".

- `vrdeproperty "H3DRedirect/Enabled=1"`

  enables the VRDE server feature: 3D redirection. To re-disable this feature, use "H3DRedirect/Enabled=".

- `vrdeproperty "Security/Method|ServerCertificate|ServerPrivateKey|CACertificate=<value>"` sets the desired security method/Path of server certificate, path of server private key, path of CA certificate, used for a connection.
  - `vrdeproperty "Security/Method=<value>"` sets the desired security method, which is used for a connection. Valid values are:
    * `Negotiate` - both Enhanced (TLS) and Standard RDP Security connections are allowed. The security method is negotiated with the client. This is the default setting.
    * `RDP` - only Standard RDP Security is accepted.
    * `TLS` - only Enhanced RDP Security is accepted. The client must support TLS.

    For details, see chapter 7.1.6, *RDP encryption*, page 109.
  - `vrdeproperty "Security/ServerCertificate=<value>"` where <value> is the absolute path of the server certificate. For details, see chapter 7.1.6, *RDP encryption*, page 109.
  - `vrdeproperty "Security/ServerPrivateKey=<value>"` where <value> is the absolute path of the server private key. For details, see chapter 7.1.6, *RDP encryption*, page 109.
  - `vrdeproperty "Security/CACertificate=<value>"` where <value> is the absolute path of the CA self signed certificate. For details, see chapter 7.1.6, *RDP encryption*, page 109.

- `vrdeproperty "Audio/RateCorrectionMode|LogPath=<value>"` sets the Audio connection mode, or Path of the audio logfile.
  - `vrdeproperty "Audio/RateCorrectionMode=<value>"` where <value> is the desired rate correction mode, allowed values are:
    * `VRDP_AUDIO_MODE_VOID` - no mode specified, use to unset any Audio mode already set.
    * `VRDP_AUDIO_MODE_RC` - rate correction mode.
    * `VRDP_AUDIO_MODE_LPF` - low pass filter mode.
    * `VRDP_AUDIO_MODE_CS` - client sync mode to prevent under/overflow of the client queue.
  - `vrdeproperty "Audio/LogPath=<value>"` where <value> is the absolute path of the Audio log file.

- `vrdevideochannelquality <percent>`: Sets the image quality for video redirection; see chapter 7.1.9, *VRDP video redirection*, page 111.

- `setvideomodehint` requests that the guest system change to a particular video mode. This requires that the Guest Additions be installed, and will not work for all guest systems.

- `screenshotpng` takes a screenshot of the guest display and saves it in PNG format.

- `videocap on|off` enables or disables recording a VM session into a WebM/VP8 file.

- `videocapscreens all|<screen ID> [<screen ID> ...]]` allows to specify which screens of the VM are being recorded. This setting cannot be changed while video capturing is enabled. Each screen is recorded into a separate file.
- `videocapfile <file>` sets the filename VirtualBox uses to save the recorded content. This setting cannot be changed while video capturing is enabled.
- `videocapres <width> <height>` sets the resolution (in pixels) of the recorded video. This setting cannot be changed while video capturing is enabled.
- `videocaprate <rate>` sets the bitrate in kilobits (kb) per second. Increasing this value makes the video look better for the cost of an increased file size. This setting cannot be changed while video capturing is enabled.
- `videocapfps <fps>` sets the maximum number of frames per second (FPS) to be recorded. Frames with a higher frequency will be skipped. Reducing this value increases the number of skipped frames and reduces the file size. This setting cannot be changed while video capturing is enabled.
- `videocapmaxtime <ms>` sets the maximum time in milliseconds the video capturing will be enabled since activation. The capturing stops when the defined time interval has elapsed. If this value is zero the capturing is not limited by time. This setting cannot be changed while video capturing is enabled.
- `videocapmaxsize <MB>` limits the maximum size of the captured video file (in MB). The capturing stops when the file size has reached the specified size. If this value is zero the capturing will not be limited by file size. This setting cannot be changed while video capturing is enabled.
- `videocapopts <key=value>[,<key=value> ...]` can be used to specify additional video capturing options. These options only are for advanced users and must be specified in a comma-separated key=value format, e.g. `foo=bar,a=b`. This setting cannot be changed while video capturing is enabled.
- The `setcredentials` operation is used for remote logons in Windows guests. For details, please refer to chapter 9.2, *Automated guest logons*, page 187.
- `teleport --host <name> --port <port>` can be used to configure a VM as a target for teleporting. <name> specifies the virtual machine name. <port> specifies the port on the virtual machine which should listen for a teleporting request from another virtual machine. It can be any free TCP/IP port number (e.g. 6000); See chapter 7.2, *Teleporting*, page 112 for an introduction.
  - `--maxdowntime <msec>`: Optional - specifies the maximum downtime (milliseconds) for the teleporting target VM.
  - `--password <password>`: Optional - indicates that the teleporting request will only succeed if the source machine specifies the same password as the one given with this command.
  - `--passwordfile <password file>`: Optional - indicates that the teleporting request will only succeed if the source machine specifies the same password as the one specified in the file given with this command. Use `stdin` to read the password from stdin.
- `plugcpu|unplugcpu <id>`: If CPU hot-plugging is enabled, this adds a virtual CPU to the virtual machines (or removes one). `<id>` specifies the index of the virtual CPU to be added or removed and must be a number from 0 to the maximum no. of CPUs configured. CPU 0 can never be removed.

- The `cpuexecutioncap <1-100>`: This operation controls how much cpu time a virtual CPU can use. A value of 50 implies a single virtual CPU can use up to 50% of a single host CPU.

- `webcam attach <path|alias> [<key=value>[;<key=value>...]]`: This operation attaches a webcam to a running VM. Specify the absolute path of the webcam on the host operating system, or use its alias (obtained by using the command: VBoxManage list webcams).

  Note that alias '.0' means default video input device on the host operating system, '.1', '.2', etc. mean first, second, etc. video input device. The device order is host-specific.

  The optional settings parameter is a ';' delimited list of name/value pairs, enabling configuration of the emulated webcam device.

  The following settings are supported:

  MaxFramerate (default no maximum limit) - this specifies the highest rate (frames/sec) at which video frames are sent to the guest. Higher frame rates increase CPU load, so this setting can be useful when there is a need to reduce CPU load. Its default 'value' is 'no maximum limit', thus enabling the guest to use all frame rates supported by the host webcam.

  MaxPayloadTransferSize (default 3060 bytes) - this specifies the maximum number of bytes the emulated webcam can send to the guest in one buffer. The default is used by some webcams. Higher values can slightly reduce CPU load, if the guest is able to use larger buffers. Note that higher MaxPayloadTransferSize values may be not supported by some guest operating systems.

- `webcam detach <path|alias>`: This operation detaches a webcam from a running VM. Specify the absolute path of the webcam on the host, or use its alias (obtained from webcam list below).

  Note the points below relating to specific Host Operating Systems:

  Windows hosts

  When the webcam device is detached from the host, the emulated webcam device is automatically detached from the guest.

  Mac OS X hosts

  OS X version 10.7 or newer is required.

  When the webcam device is detached from the host, the emulated webcam device remains attached to the guest and must be manually detached using the VBoxManage controlvm "VM name" webcam detach command.

  Linux hosts

  When the webcam is detached from the host, the emulated webcam device is automatically detached from the guest only if the webcam is streaming video. If the emulated webcam is inactive, it should be manually detached using the VBoxManage controlvm "VM name" webcam detach command.

- `webcam list`: This operation lists webcams attached to the running VM. The output is a list of absolute paths or aliases that were used for attaching the webcams to the VM using the 'webcam attach' command above.

- `addencpassword <id> <password file>|- [--removeonsuspend <yes|no>]`: This operation supplies an encrypted VM specified by <id> with the encryption password to enable a headless start. Either specify the absolute path of a password file on the host file system: <password file>, or use a '-' to instruct VBoxManage to prompt the user for the encryption password.

`--removeonsuspend <yes|no>` specifies whether to remove/keep the password from/in VM memory when the VM is suspended. If the VM has been suspended and the password has been removed, the user needs to resupply the password before the VM can be resumed. This feature is useful in cases where the user doesn't want the password to be stored in VM memory, and the VM is suspended by a host suspend event.

Note: On VirtualBox versions 5.0 and later, data stored on hard disk images can be transparently encrypted for the guest. VirtualBox uses the AES algorithm in XTS mode and supports 128 or 256 bit data encryption keys (DEK). The DEK is stored encrypted in the medium properties, and is decrypted during VM startup by supplying the encryption password.

The "VBoxManage encryptmedium" operation is used to create a DEK encrypted medium. See chapter 9.31.2, *Encrypting disk images*, page 227" for details. When starting an encrypted VM from a VirtualBox GUI app, the user will be prompted for the encryption password.

For a headless encrypted VM start, use:

VBoxManage startvm "vmname" –type headless

followed by:

VBoxManage "vmname" controlvm "vmname" addencpassword ...

to supply the encryption password required.

- `removeencpassword <id>`: This operation removes encryption password authorization for password <id> for all encrypted media attached to the VM.
- `removeallencpasswords`: This operation removes encryption password authorization for all passwords for all encrypted media attached to the VM.

8.14 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.

8.15 VBoxManage adoptstate

If you have a saved state file (`.sav`) that is separate from the VM configuration, you can use this command to "adopt" the file. This will change the VM to saved state and when you start it, VirtualBox will attempt to restore it from the saved state file you indicated. This command should only be used in special setups.

8.16 VBoxManage snapshot

This command is used to control snapshots from the command line. A snapshot consists of a complete copy of the virtual machine settings, copied at the time when the snapshot was taken, and optionally a virtual machine saved state file if the snapshot was taken while the machine was running. After a snapshot has been taken, VirtualBox creates differencing hard disk for each normal hard disk associated with the machine so that when a snapshot is restored, the contents of the virtual machine's virtual hard disks can be quickly reset by simply dropping the pre-existing differencing files.

```
VBoxManage snapshot         <uuid|vmname>
                            take <name> [--description <desc>] [--live]
                                 [--uniquename Number,Timestamp,Space,Force] |
                            delete <uuid|snapname> |
                            restore <uuid|snapname> |
                            restorecurrent |
                            edit <uuid|snapname>|--current
                                 [--name <name>]
                                 [--description <desc>] |
                            list [--details|--machinereadable]
                            showvminfo <uuid|snapname>
```

The `take` operation takes a snapshot of the current state of the virtual machine. You must supply a name for the snapshot and can optionally supply a description. The new snapshot is inserted into the snapshots tree as a child of the current snapshot and then becomes the new current snapshot. The `--description` parameter allows to describe the snapshot. If `--live` is specified, the VM will not be stopped during the snapshot creation (live snapshotting).

The `delete` operation deletes a snapshot (specified by name or by UUID). This can take a while to finish since the differencing images associated with the snapshot might need to be merged with their child differencing images.

The `restore` operation will restore the given snapshot (specified by name or by UUID) by resetting the virtual machine's settings and current state to that of the snapshot. The previous current state of the machine will be lost. After this, the given snapshot becomes the new "current" snapshot so that subsequent snapshots are inserted under the snapshot from which was restored.

The `restorecurrent` operation is a shortcut to restore the current snapshot (i.e. the snapshot from which the current state is derived). This subcommand is equivalent to using the "restore" subcommand with the name or UUID of the current snapshot, except that it avoids the extra step of determining that name or UUID.

With the `edit` operation, you can change the name or description of an existing snapshot.

The `list` operation shows all snapshots of a virtual machine.

With the `showvminfo` operation, you can view the virtual machine settings that were stored with an existing snapshot.

8.17 VBoxManage closemedium

This commands removes a hard disk, DVD or floppy image from a VirtualBox media registry.[2]

```
VBoxManage closemedium      [disk|dvd|floppy] <uuid|filename>
                            [--delete]
```

Optionally, you can request that the image be deleted. You will get appropriate diagnostics that the deletion failed, however the image will become unregistered in any case.

8.18 VBoxManage storageattach

This command attaches/modifies/removes a storage medium connected to a storage controller that was previously added with the `storagectl` command (see the previous section). The syntax is as follows:

```
VBoxManage storageattach    <uuid|vmname>
                            --storagectl <name>
                            [--port <number>]
```

[2] Before VirtualBox 4.0, it was necessary to call VBoxManage openmedium before a medium could be attached to a virtual machine; that call "registered" the medium with the global VirtualBox media registry. With VirtualBox 4.0 this is no longer necessary; media are added to media registries automatically. The "closemedium" call has been retained, however, to allow for explicitly removing a medium from a registry.

```
[--device <number>]
[--type dvddrive|hdd|fdd]
[--medium none|emptydrive|additions|
          <uuid>|<filename>|host:<drive>|iscsi]
[--mtype normal|writethrough|immutable|shareable
         readonly|multiattach]
[--comment <text>]
[--setuuid <uuid>]
[--setparentuuid <uuid>]
[--passthrough on|off]
[--tempeject on|off]
[--nonrotational on|off]
[--discard on|off]
[--hotpluggable on|off]
[--bandwidthgroup name|none]
[--forceunmount]
[--server <name>|<ip>]
[--target <target>]
[--tport <port>]
[--lun <lun>]
[--encodedlun <lun>]
[--username <username>]
[--password <password>]
[--initiator <initiator>]
[--intnet]
```

A number of parameters are commonly required; the ones at the end of the list are required only for iSCSI targets (see below).

The common parameters are:

uuid|vmname The VM UUID or VM Name. Mandatory.

--storagectl Name of the storage controller. Mandatory. The list of the storage controllers currently attached to a VM can be obtained with `VBoxManage showvminfo`; see chapter 8.5, *VBoxManage showvminfo*, page 126.

--port The number of the storage controller's port which is to be modified. Mandatory, unless the storage controller has only a single port.

--device The number of the port's device which is to be modified. Mandatory, unless the storage controller has only a single device per port.

--type Define the type of the drive to which the medium is being attached/detached/modified. This argument can only be omitted if the type of medium can be determined from either the medium given with the `--medium` argument or from a previous medium attachment.

--medium Specifies what is to be attached. The following values are supported:

- "none": Any existing device should be removed from the given slot.
- "emptydrive": For a virtual DVD or floppy drive only, this makes the device slot behaves like a removeable drive into which no media has been inserted.
- "additions": For a virtual DVD drive only, this attaches the *VirtualBox Guest Additions* image to the given device slot.
- If a UUID is specified, it must be the UUID of a storage medium that is already known to VirtualBox (e.g. because it has been attached to another virtual machine). See chapter 8.4, *VBoxManage list*, page 125 for how to list known media. This medium is then attached to the given device slot.
- If a filename is specified, it must be the full path of an existing disk image (ISO, RAW, VDI, VMDK or other), which is then attached to the given device slot.

- "host:<drive>": For a virtual DVD or floppy drive only, this connects the given device slot to the specified DVD or floppy drive on the host computer.
- "iscsi": For virtual hard disks only, this allows for specifying an iSCSI target. In this case, more parameters must be given; see below.

Some of the above changes, in particular for removeable media (floppies and CDs/DVDs), can be effected while a VM is running. Others (device changes or changes in hard disk device slots) require the VM to be powered off.

--mtype Defines how this medium behaves with respect to snapshots and write operations. See chapter 5.4, *Special image write modes*, page 84 for details.

--comment Any description that you want to have stored with this medium (optional; for example, for an iSCSI target, "Big storage server downstairs"). This is purely descriptive and not needed for the medium to function correctly.

--setuuid, --setparentuuid Modifies the UUID or parent UUID of a medium before attaching it to a VM. This is an expert option. Inappropriate use can make the medium unusable or lead to broken VM configurations if any other VM is referring to the same media already. The most frequently used variant is `--setuuid ""`, which assigns a new (random) UUID to an image. This is useful to resolve the duplicate UUID errors if one duplicated an image using file copy utilities.

--passthrough For a virtual DVD drive only, you can enable DVD writing support (currently experimental; see chapter 5.9, *CD/DVD support*, page 90).

--tempeject For a virtual DVD drive only, you can configure the behavior for guest-triggered medium eject. If this is set to "on", the eject has only temporary effects. If the VM is powered off and restarted the originally configured medium will be still in the drive.

--nonrotational This switch allows to enable the non-rotational flag for virtual hard disks. Some guests (i.e. Windows 7+) treat such disks like SSDs and don't perform disk fragmentation on such media.

--discard This switch enables the auto-discard feature for the virtual hard disks. This specifies that a VDI image will be shrunk in response to the trim command from the guest OS. The following requirements must be met:

- The disk format must be VDI.
- The size of the cleared area must be at least 1MB.
- VirtualBox will only trim whole 1MB blocks. The VDIs themselves are organized into 1MB blocks, so this will only work if the space being TRIM-med is at least a 1MB contiguous block at a 1MB boundary. On Windows, occasional defrag (with "defrag.exe /D"), or under Linux running "btrfs filesystem defrag" as a background cron job may be beneficial.

Notes: the Guest OS must be configured to issue trim command, and typically this means that the guest OS is made to 'see' the disk as an SSD. Ext4 supports -o discard mount flag; OSX probably requires additional settings. Windows ought to automatically detect and support SSDs - at least in versions 7, 8 and 10. Linux exFAT driver (courtesy of Samsung) supports the trim command.

It is unclear whether Microsoft's implementation of exFAT supports this feature, even though that file system was originally designed for flash.

Alternatively, there are ad hoc methods to issue trim, e.g. Linux fstrim command, part of util-linux package. Earlier solutions required a user to zero out unused areas, e.g. using zerofree, and explicitly compact the disk - only possible when the VM is offline.

--bandwidthgroup Sets the bandwidth group to use for the given device; see chapter 5.8, *Limiting bandwidth for disk images*, page 89.

--forceunmount For a virtual DVD or floppy drive only, this forcibly unmounts the DVD/CD/Floppy or mounts a new DVD/CD/Floppy even if the previous one is locked down by the guest for reading. Again, see chapter 5.9, *CD/DVD support*, page 90 for details.

When "iscsi" is used with the `--medium` parameter for iSCSI support – see chapter 5.10, *iSCSI servers*, page 91 –, additional parameters must or can be used:

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

--target Target name string. This is determined by the iSCSI target and used to identify the storage resource; required.

--tport TCP/IP port number of the iSCSI service on the target (optional).

--lun Logical Unit Number of the target resource (optional). Often, this value is zero.

--encodedlun Hex encoded Logical Unit Number of the target resource (optional). Often, this value is zero.

--username, --password Username and password (initiator secret) for target authentication, if required (optional).

> **Note:** Username and password are stored without encryption (i.e. in clear text) in the XML machine configuration file if no settings password is provided. When a settings password was specified the first time, the password is stored encrypted.

--initiator iSCSI Initiator (optional). Note:

Microsoft iSCSI Initiator is a system, such as a server that attaches to an IP network and initiates requests and receives responses from an iSCSI target. The SAN components in Microsoft iSCSI Initiator are largely analogous to Fibre Channel SAN components, and they include the following:/

To transport blocks of iSCSI commands over the IP network, an iSCSI driver must be installed on the iSCSI host. An iSCSI driver is included with Microsoft iSCSI Initiator.

A gigabit Ethernet adapter that transmits 1000 megabits per second (Mbps) is recommended for the connection to an iSCSI target. Like standard 10/100 adapters, most gigabit adapters use a preexisting Category 5 or Category 6E cable that. Each port on the adapter is identified by a unique IP address.

An iSCSI target is any device that receives iSCSI commands. The device can be an end node, such as a storage device, or it can be an intermediate device, such as a network bridge between IP and Fibre Channel devices. Each port on the storage array controller or network bridge is identified by one or more IP addresses

--intnet If specified, connect to the iSCSI target via Internal Networking. This needs further configuration which is described in chapter 9.9.3, *Access iSCSI targets via Internal Networking*, page 200.

8.19 VBoxManage storagectl

This command attaches/modifies/removes a storage controller. After this, virtual media can be attached to the controller with the `storageattach` command (see the next section).

The syntax is as follows:

```
VBoxManage storagectl       <uuid|vmname>
                            --name <name>
                            [--add ide|sata|scsi|floppy|sas|usb|pcie]
                            [--controller LSILogic|LSILogicSAS|BusLogic|
                                          IntelAhci|PIIX3|PIIX4|ICH6|I82078|
                                          USB|NVMe]
                            [--portcount <1-30>]
                            [--hostiocache on|off]
                            [--bootable on|off]
                            [--rename <name>]
                            [--remove]
```

where the parameters mean:

`uuid|vmname` The VM UUID or VM Name. Mandatory.

`--name` Name of the storage controller. Mandatory.

`--add` Define the type of the system bus to which the storage controller must be connected.

`--controller` Allows to choose the type of chipset being emulated for the given storage controller.

`--portcount` This determines how many ports the storage controller should support.

`--hostiocache` Configures the use of the host I/O cache for all disk images attached to this storage controller. For details, please see chapter 5.7, *Host I/O caching*, page 89.

`--bootable` Selects whether this controller is bootable.

`--rename` Sets the name of the storage controller.

`--remove` Removes the storage controller from the VM config.

8.20 VBoxManage bandwidthctl

This command creates/deletes/modifies/shows bandwidth groups of the given virtual machine:

```
VBoxManage bandwidthctl    <uuid|vmname>
                           add <name> --type disk|network --limit <megabytes per second>[k|m|g|K|M|G] |
                           set <name> --limit <megabytes per second>[k|m|g|K|M|G] |
                           remove <name> |
                           list [--machinereadable]
```

The following subcommands are available:

- `add`, creates a new bandwidth group of given type.
- `set`, modifies the limit for an existing bandwidth group.
- `remove`, destroys a bandwidth group.
- `list`, shows all bandwidth groups defined for the given VM. Use `--machinereadable` option to produce the same output, but in machine readable format. This is of the form: name="value" on a line by line basis.

The parameters mean:

uuid|vmname The VM UUID or VM Name. Mandatory.

--name Name of the bandwidth group. Mandatory.

--type Type of the bandwidth group. Mandatory. Two types are supported: `disk` and `network`. See chapter 5.8, *Limiting bandwidth for disk images*, page 89 or chapter 6.10, *Limiting bandwidth for network I/O*, page 101 for a description of a particular type.

--limit Specifies the limit for the given group. Can be changed while the VM is running. The default unit is megabytes per second. The unit can be changed by specifying one of the following suffixes: `k` for kilobits/s, `m` for megabits/s, `g` for gigabits/s, `K` for kilobytes/s, `M` for megabytes/s, `G` for gigabytes/s.

> **Note:** The network bandwidth limits apply only to the traffic being sent by virtual machines. The traffic being received by VMs is unlimited.

> **Note:** To remove a bandwidth group it must not be referenced by any disks or adapters in running VM.

8.21 VBoxManage showmediuminfo

This command shows information about a medium, notably its size, its size on disk, its type and the virtual machines which use it.

> **Note:** For compatibility with earlier versions of VirtualBox, the "showvdiinfo" command is also supported and mapped internally to the "showmediuminfo" command.

```
VBoxManage showmediuminfo     [disk|dvd|floppy] <uuid|filename>
```

The medium must be specified either by its UUID (if the medium is registered) or by its filename. Registered images can be listed by `VBoxManage list hdds`, `VBoxManage list dvds`, or `VBoxManage list floppies`, as appropriate. (see chapter 8.4, *VBoxManage list*, page 125 for more information).

8.22 VBoxManage createhd

This command creates a new medium. The syntax is as follows:

```
VBoxManage createmedium     [disk|dvd|floppy]    --filename <filename>
                            [--size <megabytes>|--sizebyte <bytes>]
                            [--diffparent <uuid>|<filename>
                            [--format VDI|VMDK|VHD] (default: VDI)
                            [--variant Standard,Fixed,Split2G,Stream,ESX]
```

where the parameters mean:

--filename <filename> Specifies a file name <filename> as an absolute path on the host file system.. Mandatory.

--size <megabytes> <megabytes> Specifies the image capacity, in 1 MB units. Optional.

--diffparent <uuid>|<filename> Specifies the differencing image parent, either as a UUID or by the absolute pathname of the file on the host file system. Useful for sharing a base box disk image among several VMs.

--format VDI|VMDK|VHD Specifies the file format for the output file. Available options are VDI, VMDK, VHD. Default is VDI. Optional.

--variant Standard,Fixed,Split2G,Stream,ESX Specifies any required file format variant(s) for the output file. It is a comma-separated list of variant flags. Not all combinations are supported, and specifying mutually incompatible flags results in an error message. Optional.

> **Note:** For compatibility with earlier versions of VirtualBox, the "createvdi" command is also supported and mapped internally to the "createmedium" command.

8.23 VBoxManage modifymedium

With the `modifymedium` command, you can change the characteristics of a disk image after it has been created:

```
VBoxManage modifymedium  [disk|dvd|floppy]    <uuid|filename>
                         [--type normal|writethrough|immutable|shareable|
                                 readonly|multiattach]
                         [--autoreset on|off]
                         [--property <name=[value]>]
                         [--compact]
                         [--resize <megabytes>|--resizebyte <bytes>]
                         [--move <full path to a new location>
```

> **Note:** For compatibility with earlier versions of VirtualBox, the "modifyvdi" and "modifyhd" commands are also supported and mapped internally to the "modifymedium" command.

The disk image to modify must be specified either by its UUID (if the medium is registered) or by its filename. Registered images can be listed by `VBoxManage list hdds` (see chapter 8.4, *VBoxManage list*, page 125 for more information). A filename must be specified as valid path, either as an absolute path or as a relative path starting from the current directory.

The following options are available:

- With the `--type` argument, you can change the type of an existing image between the normal, immutable, write-through and other modes; see chapter 5.4, *Special image write modes*, page 84 for details.
- For immutable (differencing) hard disks only, the `--autoreset on|off` option determines whether the disk is automatically reset on every VM startup (again, see chapter 5.4, *Special image write modes*, page 84). The default is "on".
- The `--compact` option, can be used to compact disk images, i.e. remove blocks that only contains zeroes. This will shrink a dynamically allocated image again; it will reduce the *physical* size of the image without affecting the logical size of the virtual disk. Compaction works both for base images and for diff images created as part of a snapshot.

For this operation to be effective, it is required that free space in the guest system first be zeroed out using a suitable software tool. For Windows guests, you can use the `sdelete` tool provided by Microsoft. Execute `sdelete -z` in the guest to zero the free disk space before compressing the virtual disk image. For Linux, use the `zerofree` utility which supports ext2/ext3 filesystems. For Mac OS X guests, use the `diskutil secureErase freespace 0 /` command line from an elevated Terminal.

Please note that compacting is currently only available for VDI images. A similar effect can be achieved by zeroing out free blocks and then cloning the disk to any other dynamically allocated format. You can use this workaround until compacting is also supported for disk formats other than VDI.

- The `--resize x` option (where x is the desired new total space in **megabytes**) allows you to change the capacity of an existing image; this adjusts the *logical* size of a virtual disk without affecting the physical size much.[3] This currently works only for VDI and VHD formats, and only for the dynamically allocated variants, and can only be used to expand (not shrink) the capacity. For example, if you originally created a 10G disk which is now full, you can use the `--resize 15360` command to change the capacity to 15G (15,360MB) without having to create a new image and copy all data from within a virtual machine. Note however that this only changes the drive capacity; you will typically next need to use a partition management tool inside the guest to adjust the main partition to fill the drive.

  The `--resizebyte x` option does almost the same thing, except that x is expressed in bytes instead of megabytes.

- The `--move <dest>` option can be used to move an image to a different location <dest> on the host file system, specified by either the relative path to the current directory or absolute path.

8.24 VBoxManage clonemedium

This command duplicates a virtual disk/DVD/floppy medium to a new medium (usually an image file) with a new unique identifier (UUID). The new image can be transferred to another host system or imported into VirtualBox again using the Virtual Media Manager; see chapter 5.3, *The Virtual Media Manager*, page 83 and chapter 5.6, *Cloning disk images*, page 88. The syntax is as follows:

```
VBoxManage clonemedium      [disk|dvd|floppy] <uuid|inputfile> <uuid|outputfile>

                            [--format VDI|VMDK|VHD|RAW|<other>]
                            [--variant Standard,Fixed,Split2G,Stream,ESX]
                            [--existing]
```

The medium to clone as well as the target image must be described either by its UUIDs (if the mediums are registered) or by its filename. Registered images can be listed by `VBoxManage list hdds` (see chapter 8.4, *VBoxManage list*, page 125 for more information). A filename must be specified as valid path, either as an absolute path or as a relative path starting from the current directory.

The following options are available:

`--format` Allow to choose a file format for the output file different from the file format of the input file.

[3] Image resizing was added with VirtualBox 4.0.

--variant Allow to choose a file format variant for the output file. It is a comma-separated list of variant flags. Not all combinations are supported, and specifying inconsistent flags will result in an error message.

--existing Perform the clone operation to an already existing destination medium. Only the portion of the source medium which fits into the destination medium is copied. This means if the destination medium is smaller than the source only a part of it is copied, and if the destination medium is larger than the source the remaining part of the destination medium is unchanged.

> **Note:** For compatibility with earlier versions of VirtualBox, the "clonevdi" and "clonehd" commands are still supported and mapped internally to the "clonehd disk" command.

8.25 VBoxManage mediumproperty

This command sets up, gets or deletes a medium property. The syntax is as follows:

```
VBoxManage mediumproperty [disk|dvd|floppy] set <uuid|filename>
                                                <property> <value>
```

- Use `<disk|dvd|floppy>` to optionally specify the type of medium: disk (hard drive), dvd or floppy.
- Use `<uuid|filename>` to supply either the uuid or absolute path of the medium/image to be encrypted.
- Use `<property>` to supply the name of the property.
- Use `<value>` to supply the property value.

```
VBoxManage mediumproperty [disk|dvd|floppy] get <uuid|filename>
                                                <property>
```

- Use `<disk|dvd|floppy>` to optionally specify the type of medium: disk (hard drive), dvd or floppy.
- Use `<uuid|filename>` to supply either the uuid or absolute path of the medium/image to be encrypted.
- Use `<property>` to supply the name of the property.

```
VBoxManage mediumproperty [disk|dvd|floppy] delete <uuid|filename>
                                                   <property>
```

- Use `<disk|dvd|floppy>` to optionally specify the type of medium: disk (hard drive), dvd or floppy.
- Use `<uuid|filename>` to supply either the uuid or absolute path of the medium/image.
- Use `<property>` to supply the name of the property.

8.26 VBoxManage encryptmedium

This command is used to create a DEK encrypted medium/image. See chapter 9.31.2, *Encrypting disk images*, page 227“ for details.

The syntax is as follows:

```
VBoxManage encryptmedium <uuid|filename>
                         [--newpassword <file|->]
                         [--oldpassword <file|->]
                         [--cipher <cipher id>]
                         [--newpasswordid <password id>]
```

- use `<uuid|filename>` to supply the uuid or absolute path of the medium/image to be encrypted.
- Use `--newpassword <file|->` to supply a new encryption password; either specify the absolute pathname of a password file on the host operating system, or `-` to prompt you for the password on the command line. Always use the `--newpasswordid` option with this option.
- use `--oldpassword <file|->` to supply any old encryption password; either specify the absolute pathname of a password file on the host operating system, or `-` to prompt you for the old password on the command line.

  Use this option to gain access to an encrypted medium/image to change its password using `--newpassword` and/or change its encryption using `--cipher`.
- Use `--cipher <cipher>` to specify the cipher to use for encryption; this can be either `AES-XTS128-PLAIN64` or `AES-AXTS256-PLAIN64`.

  Use this option to change any existing encryption on the medium/image, or setup new encryption on it for the 1st time.
- Use `--newpasswordid <password id>` to supply the new password identifier. This can be freely chosen by the user, and is used for correct identification when supplying multiple passwords during VM startup.

  If the user uses the same password when encrypting multiple images and also the same password identifier, the user needs to supply the password only once during VM startup.

8.27 VBoxManage checkmediumpwd

This command is used to check the current encryption password on a DEK encrypted medium/image. See chapter 9.31.2, *Encrypting disk images*, page 227“ for details.

The syntax is as follows:

```
VBoxManage checkmediumpwd <uuid|filename>
                                     <pwd file|->
```

- Use `<uuid|filename>` to supply the uuid or absolute path of the medium/image to be checked.
- Use `<pwd file|->` to supply the password identifier to be checked. Either specify the absolute pathname of a password file on the host operating system, or `-` to prompt you for the password on the command line.

8.28 VBoxManage convertfromraw

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

```
VBoxManage convertfromraw   <filename> <outputfile>
                            [--format VDI|VMDK|VHD]
                            [--variant Standard,Fixed,Split2G,Stream,ESX]
                            [--uuid <uuid>]
VBoxManage convertfromraw   stdin <outputfile> <bytes>
                            [--format VDI|VMDK|VHD]
                            [--variant Standard,Fixed,Split2G,Stream,ESX]
                            [--uuid <uuid>]
```

where the parameters mean:

--bytes The size of the image file, in bytes, provided through stdin.

--format Select the disk image format to create. Default is VDI. Other options are VMDK and VHD.

--variant Allow to choose a file format variant for the output file. It is a comma-separated list of variant flags. Not all combinations are supported, and specifying inconsistent flags will result in an error message.

--uuid Allow to specifiy the UUID of the output file.

The second form forces VBoxManage to read the content for the disk image from standard input (useful for using that command in a pipe).

> **Note:** For compatibility with earlier versions of VirtualBox, the "convertdd" command is also supported and mapped internally to the "convertfromraw" command.

8.29 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 5.1.22
(C) 2005-2017 Oracle Corporation
All rights reserved.

Value: 2006.01.01
```

You could retrieve the information for all keys as follows:

```
VBoxManage getextradata Fedora5 enumerate
```

To remove a key, the `setextradata` command must be run without specifying data (only the key), for example:

```
VBoxManage setextradata Fedora5 installdate
```

8.30 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:

machinefolder This specifies the default folder in which virtual machine definitions are kept; see chapter 10.1, *Where VirtualBox stores its files*, page 234 for details.

hwvirtexclusive This specifies whether VirtualBox will make exclusive use of the hardware virtualization extensions (Intel VT-x or AMD-V) of the host system's processor; see chapter 10.3, *Hardware vs. software virtualization*, page 239. If you wish to share these extensions with other hypervisors running at the same time, you must disable this setting. Doing so has negative performance implications.

vrdeauthlibrary This specifies which library to use when "external" authentication has been selected for a particular virtual machine; see chapter 7.1.5, *RDP authentication*, page 108 for details.

websrvauthlibrary This specifies which library the web service uses to authenticate users. For details about the VirtualBox web service, please refer to the separate VirtualBox SDK reference (see chapter 11, *VirtualBox programming interfaces*, page 245).

vrdeextpack This specifies which library implements the VirtualBox Remote Desktop Extension.

loghistorycount This selects how many rotated (old) VM logs are kept.

autostartdbpath This selects the path to the autostart database. See chapter 9.24, *Starting virtual machines during system boot*, page 223.

defaultfrontend This selects the global default VM frontend setting. See chapter 8.12, *VBoxManage startvm*, page 142.

logginglevel This configures the VBoxSVC release logging details.[4]

8.31 VBoxManage usbfilter add/modify/remove

```
VBoxManage usbfilter        add <index,0-N>
                          --target <uuid|vmname>global
                          --name <string>
                          --action ignore|hold (global filters only)
                         [--active yes|no (yes)]
                         [--vendorid <XXXX> (null)]
                         [--productid <XXXX> (null)]
                         [--revision <IIFF> (null)]
                         [--manufacturer <string> (null)]
                         [--product <string> (null)]
                         [--remote yes|no (null, VM filters only)]
                         [--serialnumber <string> (null)]
                         [--maskedinterfaces <XXXXXXXX>]


VBoxManage usbfilter        modify <index,0-N>
                          --target <uuid|vmname>global
                         [--name <string>]
                         [--action ignore|hold (global filters only)]
                         [--active yes|no]
                         [--vendorid <XXXX>]
                         [--productid <XXXX>]
```

[4] http://www.virtualbox.org/wiki/VBoxLogging.

```
                        [--revision <IIFF>]
                        [--manufacturer <string>]
                        [--product <string>]
                        [--remote yes|no (null, VM filters only)]
                        [--serialnumber <string>]
                        [--maskedinterfaces <XXXXXXXX>]


VBoxManage usbfilter        remove <index,0-N>
                         --target <uuid|vmname>global
```

The `usbfilter` commands are used for working with USB filters in virtual machines, or global filters which affect the whole 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 VMs access to devices that fit the filter description ("hold") or not to give them access ("ignore"). In addition, you should specify parameters to filter by. You can find the parameters for devices attached to your system using `VBoxManage list usbhost`. Finally, you can specify whether the filter should be active, 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.

The following is a list of the additional `usbfilter add` and `usbfilter modify` options, with detailed explanations on how to use them.

- `--action ignore|hold`Specifies whether devices that fit the filter description are allowed access by machines ("hold"), or have access denied ("ignore"). Applies to global filters only.
- `--active yes|no`Specifies whether the USB Filter is active or temporarily disabled. For `usbfilter create` the default is active.
- `--vendorid <XXXX>|""`Specifies a vendor ID filter - the string representation for the exact matching has the form XXXX, where X is the hex digit (including leading zeroes).
- `--productid <XXXX>|""`Specifies a product ID filter - The string representation for the exact matching has the form XXXX, where X is the hex digit (including leading zeroes).
- `--revision <IIFF>|""`Specifies a revision ID filter - the string representation for the exact matching has the form IIFF, where I is the decimal digit of the integer part of the revision, and F is the decimal digit of its fractional part (including leading and trailing zeros). Note that for interval filters, it's best to use the hex form, because the revision is stored as a 16 bit packed BCD value; so the expression int:0x0100-0x0199 will match any revision from 1.0 to 1.99 inclusive.

- `--manufacturer <string>|""`Specifies a manufacturer ID filter, as a string.
- `--product <string>|""`Specifies a product ID filter, as a string.
- `--remote yes|no""`Specifies a remote filter - indicating whether the device is physically connected to a remote VRDE client or to a local host machine. Applies to VM filters only.
- `--serialnumber <string>|""`Specifies a serial number filter, as a string.
- `--maskedinterfaces <XXXXXXXX>`Specifies a masked interface filter, for hiding one or more USB interfaces from the guest. The value is a bit mask where the set bits correspond to the USB interfaces that should be hidden, or masked off. This feature only works on Linux hosts.

8.32 VBoxManage sharedfolder add/remove

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

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.

Parameters are:

- `<uuid|vmname>` Specifies the UUID or name of the VM whose guest operating system will be sharing folders with the host computer. Mandatory.
- `--name <name>` Specifies the name of the share. Each share has a unique name within the namespace of the host operating system. Mandatory.
- `-hostpath <hostpath>` Specifies the absolute path on the host operating system of the folder/directory to be shared with the guest operating system. Mandatory.
- `-transient` Specifies that the share is 'transient', meaning that it can be added and removed at runtime and does not persist after the VM has stopped. Optional.
- `-readonly` Specifies that the share has only read-only access to files at the host path.

  By default, shared folders have read/write access to the files at the host path. More specifically, on Linux distros - shared folders are mounted with 770 io permissions with root user and vboxsf as the group, and using this option the io permissions change to 700. Optional.
- `-automount` Specifies that the share will be automatically mounted. On Linux distros, this will be to either /media/USER/sf_<name> or /media/sf_<name> - depending on your guest OS. Where <name> is the share name. Optional.

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

This command allows you to delete shared folders on the host computer shares with the guest operating systems. For this, the guest systems must have a version of the VirtualBox Guest Additions installed which supports this functionality.

Parameters are:

- <uuid|vmname> Specifies the UUID or name of the VM whose guest operating system is sharing folders with the host computer. Mandatory.
- --name <name> Specifies the name of the share to be removed. Each share has a unique name within the namespace of the host operating system. Mandatory.
- -transient Specifies that the share is 'transient', meaning that it can be added and removed at runtime and does not persist after the VM has stopped. Optional.

Shared folders are described in detail in chapter 4.3, *Shared folders*, page 68.

8.33 VBoxManage guestproperty

The "guestproperty" commands allow you to get or set properties of a running virtual machine. Please see chapter 4.7, *Guest properties*, page 74 for an introduction. As explained there, guest properties are arbitrary key/value string pairs which can be written to and read from by either the guest or the host, so they can be used as a low-volume communication channel for strings, provided that a guest is running and has the Guest Additions installed. In addition, a number of values whose keys begin with "/VirtualBox/" are automatically set and maintained by the Guest Additions.

The following subcommands are available (where <vm>, in each case, can either be a VM name or a VM UUID, as with the other VBoxManage commands):

- enumerate <vm> [--patterns <pattern>]: This lists all the guest properties that are available for the given VM, including the value. This list will be very limited if the guest's service process cannot be contacted, e.g. because the VM is not running or the Guest Additions are not installed.

  If --patterns <pattern> is specified, it acts as a filter to only list properties that match the given pattern. The pattern can contain the following wildcard characters:

  - * (asterisk): represents any number of characters; for example, "/VirtualBox*" would match all properties beginning with "/VirtualBox".
  - ? (question mark): represents a single arbitrary character; for example, "fo?" would match both "foo" and "for".
  - | (pipe symbol): can be used to specify multiple alternative patterns; for example, "s*|t*" would match anything starting with either "s" or "t".

- get <vm> <property> : This retrieves the value of a single property only. If the property cannot be found (e.g. because the guest is not running), this will print

  ```
  No value set!
  ```

- set <vm> <property> [<value> [--flags <flags>]]: This allows you to set a guest property by specifying the key and value. If <value> is omitted, the property is deleted. With --flags you can optionally specify additional behavior (you can combine several by separating them with commas):

  - TRANSIENT: the value will not be stored with the VM data when the VM exits;
  - TRANSRESET: the value will be deleted as soon as the VM restarts and/or exits;
  - RDONLYGUEST: the value can only be changed by the host, but the guest can only read it;
  - RDONLYHOST: reversely, the value can only be changed by the guest, but the host can only read it;
  - READONLY: a combination of the two, the value cannot be changed at all.

- `wait <vm> <pattern> --timeout <timeout>`: This waits for a particular value described by "pattern" to change or to be deleted or created. The pattern rules are the same as for the "enumerate" subcommand above.
- `delete <vm> <property>` : Deletes a formerly set guest property.

8.34 VBoxManage guestcontrol

The `guestcontrol` commands enable control of the guest from the host. Please see chapter 4.8, *Guest control*, page 76 for an introduction.

guestcontrol has two sets of subcommands. The first set requires guest credentials to be specified, the second does not.

The first set of subcommands is of the form:

```
VBoxManage guestcontrol <uuid|vmname> <sub-command>
            [--username <name> ]
            [--passwordfile <file> | --password <password>]
            [--domain <domain> ]
            [-v|--verbose] [-q|quiet] ...
```

The "common-options" are:

```
[--username <name> ]
[--passwordfile <file> | --password <password>]
[--domain <domain> ]
[-v|--verbose] [-q|quiet]
```

Where details of the common options for the first set of subcommands are:

`<uuid|vmname>` Specifies the VM UUID or VM name. Mandatory.

`--username <name>` Specifies the user name on guest OS under which the process should run. This user name must already exist on the guest OS. If unspecified, the host user name is used. Optional

`--passwordfile <file>|--password` Specifies the absolute path on guest file system of password file containing the password for the specified user account or password for the specified user account. Optional. If both are omitted, empty password is assumed.

`--domain <domain>` User domain for Windows guests. Optional.

`-v|--verbose` Makes the subcommand execution more verbose. Optional

`-q|--quiet` Makes the subcommand execution quieter. Optional.

The first set of subcommands:

- **`run`** Executes a guest program - forwarding stdout, stderr and stdin to/from the host until it completes.

  ```
  VBoxManage guestcontrol <uuid|vmname> run [common-options]
              --exe <path to executable> [--timeout <msec>]
             [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
             [--ignore-operhaned-processes] [--profile]
             [--no-wait-stdout|--wait-stdout]
             [--no-wait-stderr|--wait-stderr]
             [--dos2unix] [--unix2dos]
              -- <program/arg0> [argument1] ... [argumentN]]
  ```

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--exe <path to executable> Specifies the absolute path of the executable on the guest OS file system. Mandatory. e.g.: `C:\Windows\System32\calc.exe`.

--timeout <msec> Specifies the maximum time (microseconds) that the executable can run, during which VBoxManage receives its output. Optional. If unspecified, VBoxManage waits indefinitely for the process to end, or an error occurs.

-E|--putenv <NAME>=<VALUE> Sets/modifies/unsets environment variable(s) in the environment in which the program will run. Optional.

The guest process is created with the standard default guest OS environment. Use this option to modify that default environment. To set/modify a variable use: `<NAME>=<VALUE>`. To unset a variable use: `<NAME>=`

Any spaces in names/values should be enclosed by quotes.

To set/modify/unset multiple variables, use multiple instances of the `--E|--putenv` option.

--unquoted-args Disables escaped double quoting (e.g. \“fred\“) on arguments passed to the executed program. Optional.

--ignore-operhaned-processes Ignore orphaned processes. Not yet implemented. Optional.

--profile Use Profile. Not yet implemented. Optional.

--no-wait-stdout|--wait-stdout Does not wait/waits until the guest process ends and receives its exit code and reason/flags. In the case of –wait-stdout - while the process runs, VBoxManage receives its stdout. Optional.

--no-wait-stderr|--wait-stderr Does not wait/waits until the guest process ends and receives its exit code and reason/flags. In case of –wait-stderr - while the process runs, VBoxManage receives its stderr. Optional.

--dos2unix Converts output from DOS/Windows guests to UNIX/Linux-compatible line endings (CR + LF → LF). Not yet implemented. Optional.

--unix2dos Converts output from a UNIX/Linux guests to DOS/Windows-compatible line endings (LF → CR + LF). Not yet implemented. Optional.

[-- <program/arg0> [<argument1>] ... [<argumentN>]] Specifies program name, followed by one or more arguments to pass to the program. Optional.

Note: Any spaces in arguments should be enclosed by quotes.

> **Note:** On Windows there are certain limitations for graphical applications; please see chapter 14, *Known limitations*, page 269 for more information.

Examples:

```
VBoxManage --nologo guestcontrol "My VM" execute --image "/bin/ls"
          --username foo --passwordfile bar.txt --wait-exit --wait-stdout -- -l /usr

VBoxManage --nologo guestcontrol "My VM" execute --image "c:\\windows\\system32\\ipconfig.exe"
          --username foo --passwordfile bar.txt --wait-exit --wait-stdout
```

Note that the double backslashes in the second example are only required on Unix hosts.

> **Note:** For certain commands a user name of an existing user account on the guest must be specified; anonymous executions are not supported for security reasons. A user account password, however, is optional and depends on the guest's OS security policy or rules. If no password is specified for a given user name, an empty password will be used. On certain OSes like Windows the security policy may needs to be adjusted in order to allow user accounts with an empty password set. Also, global domain rules might apply and therefore cannot be changed.

Starting at VirtualBox 4.1.2 guest process execution by default is limited to serve up to 5 guest processes at a time. If a new guest process gets started which would exceed this limit, the oldest not running guest process will be discarded in order to be able to run that new process. Also, retrieving output from this old guest process will not be possible anymore then. If all 5 guest processes are still active and running, starting a new guest process will result in an appropriate error message.

To raise or lower the guest process execution limit, either the guest property `/VirtualBox/GuestAdd/VBoxService/--control-procs-max-kept` or VBoxService' command line by specifying `--control-procs-max-kept` needs to be modified. A restart of the guest OS is required afterwards. To serve unlimited guest processes, a value of `0` needs to be set (not recommended).

- **`start`** Executes a guest program until it completes.

```
VBoxManage guestcontrol <uuid|vmname> start [common-options]
           [--exe <path to executable>] [--timeout <msec>]
           [-E|--putenv <NAME>[=<VALUE>]] [--unquoted-args]
           [--ignore-operhaned-processes] [--profile]
            -- <program/arg0> [argument1] ... [argumentN]]
```

Where the options are:

`<uuid|vmname>` Specifies the VM UUID or VM name. Mandatory.

`--exe <path to executable>` Specifies the absolute path of the executable on the guest OS file system. Mandatory. e.g.: `C:\Windows\System32\calc.exe`

`--timeout <msec>` Specifies the maximum time (microseconds) that the executable can run. Optional. If unspecified, VBoxManage waits indefinitely for the process to end, or an error occurs.

`-E|--putenv <NAME>=<VALUE>` Sets/modifies/unsets environment variable(s) in the environment in which the program will run. Optional.

The guest process is created with the standard default guest OS environment. Use this option to modify that default environment. To set/modify a variable use: `<NAME>=<VALUE>`. To unset a variable use: `<NAME>=`

Any spaces in names/values should be enclosed by quotes.

To set/modify/unset multiple variables, use multiple instances of the `--E|--putenv` option.

`--unquoted-args` Disables escaped double quoting (e.g. \"fred\") on arguments passed to the executed program. Optional.

`--ignore-operhaned-processes` Ignores orphaned processes. Not yet implemented. Optional.

`--profile` Use a profile. Not yet implemented. Optional.

`[-- <program/arg0> [<argument1>] ... [<argumentN>]]` Specifies program name, followed by one or more arguments to pass to the program. Optional.

Note: Any spaces in arguments should be enclosed by quotes.

> **Note:** On Windows there are certain limitations for graphical applications; please see chapter 14, *Known limitations*, page 269 for more information.

Examples:

```
VBoxManage --nologo guestcontrol "My VM" execute --image "/bin/ls"
          --username foo --passwordfile bar.txt --wait-exit --wait-stdout -- -l /usr
```

```
VBoxManage --nologo guestcontrol "My VM" execute --image "c:\\windows\\system32\\ipconfig.exe"
          --username foo --passwordfile bar.txt --wait-exit --wait-stdout
```

Note that the double backslashes in the second example are only required on Unix hosts.

> **Note:** For certain commands a user name of an existing user account on the guest must be specified; anonymous executions are not supported for security reasons. A user account password, however, is optional and depends on the guest's OS security policy or rules. If no password is specified for a given user name, an empty password will be used. On certain OSes like Windows the security policy may needs to be adjusted in order to allow user accounts with an empty password set. Also, global domain rules might apply and therefore cannot be changed.

Starting at VirtualBox 4.1.2 guest process execution by default is limited to serve up to 5 guest processes at a time. If a new guest process gets started which would exceed this limit, the oldest not running guest process will be discarded in order to be able to run that new process. Also, retrieving output from this old guest process will not be possible anymore then. If all 5 guest processes are still active and running, starting a new guest process will result in an appropriate error message.

To raise or lower the guest process execution limit, either the guest property `/VirtualBox/GuestAdd/VBoxService/--control-procs-max-kept` or VBoxService' command line by specifying `--control-procs-max-kept` needs to be modified. A restart of the guest OS is required afterwards. To serve unlimited guest processes, a value of `0` needs to be set (not recommended).

- **`copyfrom`** Copies files from the guest to the host file system. (Note - only with Guest Additions 4.0 or later installed).

```
VBoxManage guestcontrol <uuid|vmname> copyfrom [common-options]
          [--dryrun] [--follow] [--R|recursive]
           --target-directory <host-dst-dir>
           <guest-src0> [<guest-src1> [...]]
```

Where the parameters are:

`<uid|vmname>` Specifies the VM UUID or VM name. Mandatory.

`--dryrun` Instructs VBoxManage to perform a dry run instead of an actual file copying operation. Optional.

`--follow` Enables symlink following on the guest file system. Optional.

`-R|--recursive` Enables recursive copying of files/directories from the specified guest file system directory. Optional.

`--target-directory <host-dst-dir>` Specifies the absolute path of the host file system destination directory. Mandatory. e.g. `C:\Temp`.

`<guest-src0> [<guest-src1> [...]]` Specifies the absolute path(s) of guest file system file(s) to be copied. Mandatory. e.g. `C:\Windows\System32\calc.exe`. Wildcards can be used in the expression(s), e.g. `C:\Windows\System*\*.dll`.

- **copyto** Copies files from the host to the guest file system. (Note - only with Guest Additions 4.0 or later installed).

```
VBoxManage guestcontrol <uuid|vmname> copyto [common-options]
           [--dryrun] [--follow] [--R|recursive]
            --target-directory <guest-dst>
            <host-src0> [<host-src1> [...]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--dryrun Instructs VBoxManage to perform a dry run instead of an actual file copying operation. Optional.

--follow Enables symlink following on the host file system. Optional.

-R|--recursive Enables recursive copying of files/directories from the specified host file system directory(ies). Optional.

--target-directory <guest-dst> Specifies the absolute path of the guest file system destination directory. Mandatory. e.g. `C:\Temp`.

<host-src0> [<host-src1> [...]] Specifies the absolute path(s) of host file system file(s) to be copied. Mandatory. e.g. `C:\Windows\System32\calc.exe`. Wildcards can be used in the expression(s), e.g. `C:\Windows\System*\*.dll`.

- **md|mkdir|createdir|createdirectory** Creates one or more directory(ies) on the guest file system. (Note - only with Guest Additions 4.0 or later installed).

```
VBoxManage guestcontrol <uuid|vmname>  md|mkdir|createdir|createdirectory [common-options]
            [--parents] [--mode <mode>]
            <guest-dir0> [<guest-dir1> [...]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--parents Creates any absent parent directory(ies) of the specified directory. Optional.

e.g. If specified directory is `D:\Foo\Bar` and `D:\Foo` is absent, it will be created. In such a case, had the `--parents` option not been used, this command would have failed.

--mode <mode> Specifies the permission mode on the specified directory(ies) (and any parents, where `--parents` option used). Currently octal modes (e.g. `0755`) only are supported.

<guest-dir0> [<guest-dir1> [...]] Specifies list of absolute path(s) of directory(ies) to be created on guest file system. Mandatory. e.g. `D:\Foo\Bar`.

All parent directories must already exist unless switch `--parents` used. (e.g. in the above example `D:\Foo`). The specified user must have sufficient rights to create the specified directory(ies), and any parents that need to be created.

- **rmdir|removedir|removedirectory** Deletes specified guest file system directories. (Only with installed Guest Additions 4.3.2 and later).

```
VBoxManage guestcontrol <uuid|vmname> rmdir|removedir|removedirectory [common-options]
           [--recursive|-R]
            <guest-dir0> [<guest-dir1> [...]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--recursive Recursively removes directories and contents. Optional.

<guest-dir0> [<guest-dir1> [...]] Specifies list of the absolute path(s) of directory(ies) to be deleted on guest file system. Mandatory. Wildcards are allowed. e.g. `D:\Foo\*Bar`. The specified user must have sufficient rights to delete the specified directory(ies).

- **rm|removefile** Deletes specified files on the guest file system. (Only with installed Guest Additions 4.3.2 and later).

```
VBoxManage guestcontrol <uuid|vmname> rm|removefile [common-options]
           [-f|--force]
            <guest-file0> [<guest-file1> [...]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

-f|--force Enforce operation (override any requests for confirmations). Optional.

<guest-file0> [<guest-file1> [...]] Specifies list of absolute path(s) of file(s) to be deleted on guest file system. Mandatory. Wildcards are allowed. e.g. `D:\Foo\Bar\text*.txt`. The specified user should have sufficient rights to delete the specified file(s).

- **mv|move|ren|rename** This subcommand renames file(s) and/or directory(ies) on the guest file system. (Only with installed Guest Additions 4.3.2 and later).

```
VBoxManage guestcontrol <uuid|vmname> mv|move|ren|rename [common-options]
           <guest-source0> [<guest-source1> [...]] <guest-dest>
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

<guest-source0> [<guest-source1> [...]] Specifies absolute path(s) of file(s) and/or single directory to be moved/renamed on guest file system. Mandatory. Wildcards are allowed in file names(s). The specified user should have sufficient rights to access the specified file(s).

<dest> Specifies the absolute path of the destination file/directory to which the file(s) are to be moved. Mandatory. If only one file to be moved, <dest> can be file or directory, else it must be a directory. The specified user must have sufficient rights to access the destination file/directory.

- **mktemp|createtemp|createtemporary** Creates a temporary file/directory on the guest file system, to assist subsequent copying of files from the host to the guest file systems. By default, the file/directory is created in the guest's platform specific temp directory. Not currently supported. (Only with installed Guest Additions 4.2 and later).

```
VBoxManage guestcontrol <uuid|vmname> mktemp|createtemp|createtemporary [common-options]
           [--directory] [--secure] [--mode <mode>] [--tmpdir <directory>]
            <template>
```

The parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--directory Creates a temporary directory instead of a file, specified by the <template> parameter. Optional.

--secure Enforces secure file/directory creation. Optional. The permission mode is set to `0755`. Operation fails if it cannot be performed securely.

--mode <mode> Specifies the permission mode of the specified directory. Optional. Currently only octal modes (e.g. `0755`) are supported.

--tmpdir <directory> Specifies the absolute path of the directory on the guest file system into which the file/directory specified in will be created. Optional. If unspecified, the platform-specific temp directory is used.

<template> Specifies a file name without a directory path, containing at least one sequence comprising three consecutive 'X' characters, or ending in 'X'. Mandatory.

- **stat** Displays file or file system status(es) on the guest.

```
VBoxManage guestcontrol <uuid|vmname> stat [common-options]
           <file0> [<file1> [...]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

<file0> [<file1> [...]] Specifies absolute path(s) of file(s) and/or file system(s) on guest file system. Mandatory. e.g. `/home/foo/a.out`. The specified user should have sufficient rights to access the specified file(s)/file system(s).

The second set of subcommands is of the form:

```
VBoxManage guestcontrol <uuid|vmname> <sub-command>
           [-v|--verbose] [-q|quiet] ...
```

The "common-options" are:

```
[-v|--verbose] [-q|--quiet]
```

Where details of the common options for the second set of subcommands are:

-v|--verbose Makes the sub-command execution more verbose. Optional.

-q|--quiet Makes the sub-command execution quieter. Optional.

The second set of subcommands:

- **list** Lists guest control configuration and status data, e.g. open guest sessions, guest processes and files.

```
VBoxManage guestcontrol <uuid|vmname> list [common-opts]
           <all|sessions|processes|files>
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

all|sessions|processes|files Indicates whether to list all available data or guest sessions, processes or files. Mandatory.

- **closeprocess** Terminates guest processes specified by PID(s))running in guest session(s), specified by the session ID or name(s).

```
VBoxManage guestcontrol <uuid|vmname> closeprocess [common-options]
           --session-id <ID> | --session-name <name or pattern>
           <PID0> [<PID1> [...]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--session-id <ID> Specifies the guest session by its ID. Optional.

--session-name <name or pattern> Specifies the guest session by its name, or multiple sessions using a pattern containing wildcards. Optional.

<PID0> [<PID1> [...]] Specifies a list of process identifiers (PIDs) of guest processes to be terminated. Mandatory.

- **closesession** Closes specified guest sessions, specified either by session ID or name.

```
VBoxManage guestcontrol <uuid|vmname> closesession [common-options]
           --session-id <ID> | --session-name <name or pattern> | --all
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--session-id <ID> Specifies the guest session to be closed by ID. Optional.

--session-name <name or pattern> Specifies the guest session to be closed by name. Optional. Multiple sessions can be specified by using a pattern containing wildcards.

--all Close all guest sessions. Optional.

- **updatega|updateadditions|updateguestadditions** Ugrades Guest Additions already installed on the guest. (Only already installed Guest Additions 4.0 and later).

```
VBoxManage guestcontrol <uuid|vmname> updatega|updateadditions|updateguestadditions [common-options]
           [--source <New .ISO path>]
           [--wait-start]
           [-- <argument0> [<argument1> [...]]]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

--source <New .ISO path> Specifies the absolute path on guest file system of the .ISO file for Guest Additions update. Mandatory.

--wait-start Indicates that VBoxManage starts the usual updating process on the guest and then waits until the actual Guest Additions updating begins, at which point VBoxManage self-terminates. Optional.

Default behavior is that VBoxManage waits for completion of the Guest Additions update before terminating. Use of this option is sometimes necessary, as a running VBoxManage can affect the interaction between the installer and the guest OS.

[-- <argument0> [<argument1> [...]]] Specifies optional command line arguments to be supplied to the Guest Additions updater. Useful for retrofitting features which are not currently installed.

Arguments containing spaces should be enclosed by quotes.

- **watch** This subcommand prints current guest control activity.

```
VBoxManage guestcontrol <uuid|vmname> watch [common-options]
```

Where the parameters are:

<uuid|vmname> Specifies the VM UUID or VM name. Mandatory.

8.35 VBoxManage metrics

This command supports monitoring the usage of system resources. Resources are represented by various metrics associated with the host system or a particular VM. For example, the host system has a `CPU/Load/User` metric that shows the percentage of time CPUs spend executing in user mode over a specific sampling period.

Metric data is collected and retained internally; it may be retrieved at any time with the `VBoxManage metrics query` subcommand. The data is available as long as the background

VBoxSVC process is alive. That process terminates shortly after all VMs and frontends have been closed.

By default no metrics are collected at all. Metrics collection does not start until VBoxManage metrics setup is invoked with a proper sampling interval and the number of metrics to be retained. The interval is measured in seconds. For example, to enable collecting the host processor and memory usage metrics every second and keeping the 5 most current samples, the following command can be used:

```
VBoxManage metrics setup --period 1 --samples 5 host CPU/Load,RAM/Usage
```

Metric collection can only be enabled for started VMs. Collected data and collection settings for a particular VM will disappear as soon as it shuts down. Use VBoxManage metrics list subcommand to see which metrics are currently available. You can also use --list option with any subcommand that modifies metric settings to find out which metrics were affected.

Note that the VBoxManage metrics setup subcommand discards all samples that may have been previously collected for the specified set of objects and metrics.

To enable or disable metrics collection without discarding the data VBoxManage metrics enable and VBoxManage metrics disable subcommands can be used. Note that these subcommands expect metrics, not submetrics, like CPU/Load or RAM/Usage as parameters. In other words enabling CPU/Load/User while disabling CPU/Load/Kernel is not supported.

The host and VMs have different sets of associated metrics. Available metrics can be listed with VBoxManage metrics list subcommand.

A complete metric name may include an aggregate function. The name has the following form: Category/Metric[/SubMetric][:aggregate]. For example, RAM/Usage/Free:min stands for the minimum amount of available memory over all retained data if applied to the host object.

Subcommands may apply to all objects and metrics or can be limited to one object or/and a list of metrics. If no objects or metrics are given in the parameters, the subcommands will apply to all available metrics of all objects. You may use an asterisk ("*") to explicitly specify that the command should be applied to all objects or metrics. Use "host" as the object name to limit the scope of the command to host-related metrics. To limit the scope to a subset of metrics, use a metric list with names separated by commas.

For example, to query metric data on the CPU time spent in user and kernel modes by the virtual machine named "test", you can use the following command:

```
VBoxManage metrics query test CPU/Load/User,CPU/Load/Kernel
```

The following list summarizes the available subcommands:

list This subcommand shows the parameters of the currently existing metrics. Note that VM-specific metrics are only available when a particular VM is running.

setup This subcommand sets the interval between taking two samples of metric data and the number of samples retained internally. The retained data is available for displaying with the query subcommand. The --list option shows which metrics have been modified as the result of the command execution.

enable This subcommand "resumes" data collection after it has been stopped with disable subcommand. Note that specifying submetrics as parameters will not enable underlying metrics. Use --list to find out if the command did what was expected.

disable This subcommand "suspends" data collection without affecting collection parameters or collected data. Note that specifying submetrics as parameters will not disable underlying metrics. Use --list to find out if the command did what was expected.

query This subcommand retrieves and displays the currently retained metric data.

> **Note:** The `query` subcommand does not remove or "flush" retained data. If you query often enough you will see how old samples are gradually being "phased out" by new samples.

`collect` This subcommand sets the interval between taking two samples of metric data and the number of samples retained internally. The collected data is displayed periodically until Ctrl-C is pressed unless the `--detach` option is specified. With the `--detach` option, this subcommand operates the same way as `setup` does. The `--list` option shows which metrics match the specified filter.

8.36 VBoxManage natnetwork

NAT networks use the Network Address Translation (NAT) service - which works in a similar way to a home router. It groups systems using it into a network and prevents outside systems from directly accessing those inside, while letting systems inside communicate with each other and outside systems using TCP and UDP over IPv4 and IPv6.

A NAT service is attached to an internal network. Virtual machines to make use of one should be attached to it. The name of an internal network is chosen when the NAT service is created, and the internal network will be created if it does not already exist. An example command to create a NAT network:

```
VBoxManage natnetwork add --netname natnet1 --network "192.168.15.0/24" --enable
```

Here, "natnet1" is the name of the internal network to be used and "192.168.15.0/24" is the network address and mask of the NAT service interface. By default, in this static configuration - the gateway will be assigned the address 192.168.15.1 (the address after the interface address), though this is subject to change.

To add a DHCP server to the NAT network after creation:

```
VBoxManage natnetwork modify --netname natnet1 --dhcp on
```

Below are the subcommands for **`VBoxManage natnetwork`**

```
VBoxManage natnetwork add --netname <name>
                        [--network <network>]
                        [--enable|--disable]
                        [--dhcp on|off]
                        [--port-forward-4 <rule>]
                        [--loopback-4 <rule>]
                        [--ipv6 on|off]
                        [--port-forward-6 <rule>]
                        [--loopback-6 <rule>]
```

`VBoxManage natnetwork add` Creates a new internal network interface, and adds a NAT network service. This command is a prerequisite for enabling attachment of VMs to the NAT network. Parameters:

`--netname <name>` Where <name> is the name of the new internal network interface on the host OS.

`--network <network>` Where <network> specifies the static(default)/DHCP network address and mask of the NAT service interface.

`--enable|--disable` Enables/disables the NAT network service.

--dhcp on|off Enables/disables DHCP server specified by –netname; its use also indicates that it is a DHCP server.

--port-forward-4 <rule> Enables IPv4 port forwarding, rule specified by <rule>.

--loopback-4 <rule> Enables IPv4 loopback interface, rule specified by <rule>.

--ipv6 on|off Enables/disables IPv6 (default is IPv4, disables gives IPv4).

--port-forward-6 <rule> Enables IPv6 port forwarding, rule specified by <rule>.

--loopback-6 <rule> Enables IPv6 loopback interface, rule specified by <rule>.

```
VBoxManage natnetwork remove --netname <name>
```

VBoxManage natnetwork remove Removes a NAT network service, parameters:

--netname <name> Where <name> specifies an existing NAT network service. Does not remove any DHCP server enabled on the network.

```
VBoxManage natnetwork modify --netname <name>
                            [--network <network>]
                            [--enable|--disable]
                            [--dhcp on|off]
                            [--port-forward-4 <rule>]
                            [--loopback-4 <rule>]
                            [--ipv6 on|off]
                            [--port-forward-6 <rule>]
                            [--loopback-6 <rule>]
```

VBoxManage natnetwork modify Modifies an existing NAT network service, parameters:

--netname <name> Where <name> specifies an existing NAT network service.

--network <network> Where <network> specifies the new static(default)/DHCP network address and mask of the NAT service interface.

--enable|--disable Enables/disables the NAT network service.

--dhcp on|off Enables (and if absent, adds)/disables (if any) DHCP server.

--port-forward-4 <rule> Enables IPv4 port forwarding, rule specified by <rule>.

--loopback-4 <rule> Enables IPv4 loopback interface, rule specified by <rule>.

--ipv6 on|off Enables/disables IPv6 (default is IPv4, disables gives IPv4).

--port-forward-6 <rule> Enables IPv6 port forwarding, rule specified by <rule>.

--loopback-6 <rule> Enables IPv6 loopback interface, rule specified by <rule>.

```
VBoxManage natnetwork start --netname <name>
```

VBoxManage natnetwork start Starts specified NAT network service and any associated DHCP server, parameters:

--netname <name> Where <name> specifies an existing NAT network service.

```
VBoxManage natnetwork stop --netname <name>
```

VBoxManage natnetwork stop Stops specified NAT network service and any DHCP server, parameters:

--netname <name> Where <name> specifies an existing NAT network service.

```
VBoxManage natnetwork list [<pattern>]
```

VBoxManage natnetwork list Lists all NAT network services with optional filtering, parameters:

[<pattern>] Where <pattern> is optional filtering pattern.

8.37 VBoxManage hostonlyif

With "hostonlyif" you can change the IP configuration of a host-only network interface. For a description of host-only networking, please refer to chapter 6.7, *Host-only networking*, page 99. Each host-only interface is identified by a name and can either use the internal DHCP server or a manual IP configuration (both IP4 and IP6).

The following list summarizes the available subcommands:

ipconfig "<name>" Configure a hostonly interface

create Creates a new vboxnet<N> interface on the host OS. This command is essential before you can attach VMs to host-only network.

remove vboxnet<N> Removes a vboxnet<N> interface from the host OS.

8.38 VBoxManage dhcpserver

The "dhcpserver" commands allow you to control the DHCP server that is built into VirtualBox. You may find this useful when using internal or host-only networking. (Theoretically, you can enable it for a bridged network as well, but that will likely cause conflicts with other DHCP servers in your physical network.)

Use the following command line options:

- If you use internal networking for a virtual network adapter of a virtual machine, use `VBoxManage dhcpserver add --netname <network_name>`, where `<network_name>` is the same network name you used with `VBoxManage modifyvm <vmname> --intnet<X> <network_name>`.
- If you use host-only networking for a virtual network adapter of a virtual machine, use `VBoxManage dhcpserver add --ifname <hostonly_if_name>` instead, where `<hostonly_if_name>` is the same host-only interface name you used with `VBoxManage modifyvm <vmname> --hostonlyadapter<X> <hostonly_if_name>`.

  Alternatively, you can also use the `--netname` option as with internal networks if you know the host-only network's name; you can see the names with `VBoxManage list hostonlyifs` (see chapter 8.4, *VBoxManage list*, page 125 above).

The following additional parameters are required when first adding a DHCP server:

- With `--ip`, specify the IP address of the DHCP server itself.
- With `--netmask`, specify the netmask of the network.
- With `--lowerip` and `--upperip`, you can specify the lowest and highest IP address, respectively, that the DHCP server will hand out to clients.

Finally, you must specify `--enable` or the DHCP server will be created in the disabled state, doing nothing.

After this, VirtualBox will automatically start the DHCP server for given internal or host-only network as soon as the first virtual machine which uses that network is started.

Reversely, use `VBoxManage dhcpserver remove` with the given `--netname <network_name>` or `--ifname <hostonly_if_name>` to remove the DHCP server again for the given internal or host-only network.

To modify the settings of a DHCP server created earlier with `VBoxManage dhcpserver add`, you can use `VBoxManage dhcpserver modify` for a given network or host-only interface name. This has the same parameters as `VBoxManage dhcpserver add`.

8.39 VBoxManage usbdevsource

The "usbdevsource" commands enables you to add and remove USB devices globally.

The following command adds a USB device.

```
VBoxManage usbdevsource add <source name>
                           --backend <backend>
                           --address <address>
```

Where the command line options are:

- <source name> specifies the ID of the 'source' USB device to be added. Mandatory.
- –backend <backend> specifies the USB proxy service backend to use. Mandatory.
- –address <address> specifies the backend specific address. Mandatory.

The following command removes a USB device.

```
VBoxManage usbdevsource remove <source name>
```

Where the command line options are:

- <source name> specifies the ID of the 'source' USB device to be removed. Mandatory.

8.40 VBoxManage debugvm

Introspection and guest debugging.

Synopsis

```
VBoxManage debugvm <uuid|vmname> dumpvmcore [--filename=name]
VBoxManage debugvm <uuid|vmname> info <item> [args...]
VBoxManage debugvm <uuid|vmname> injectnmi
VBoxManage debugvm <uuid|vmname> log [[--release] | [--debug]] [group-settings...]
VBoxManage debugvm <uuid|vmname> logdest [[--release] | [--debug]] [destinations...]
VBoxManage debugvm <uuid|vmname> logflags [[--release] | [--debug]] [flags...]
VBoxManage debugvm <uuid|vmname> osdetect
VBoxManage debugvm <uuid|vmname> osinfo
VBoxManage debugvm <uuid|vmname> osdmesg [--lines=lines]
VBoxManage debugvm <uuid|vmname> getregisters [--cpu=id] [reg-set.reg-name...]
VBoxManage debugvm <uuid|vmname> setregisters [--cpu=id] [reg-set.reg-name=value...]
```

```
VBoxManage debugvm <uuid|vmname> show [[--human-readable] | [--sh-export] |
    [--sh-eval] | [--cmd-set]] [settings-item...]
VBoxManage debugvm <uuid|vmname> stack [--cpu=id]
VBoxManage debugvm <uuid|vmname> statistics [--reset] [--descriptions] [--pattern=pattern]
```

Description

The "debugvm" commands are for experts who want to tinker with the exact details of virtual machine execution. Like the VM debugger described in chapter 12.1.3, *The built-in VM debugger*, page 247, these commands are only useful if you are very familiar with the details of the PC architecture and how to debug software.

Common options

The subcommands of `debugvm` all operate on a running virtual machine:

uuid|vmname
: Either the UUID or the name (case sensitive) of a VM.

debugvm dumpvmcore

```
VBoxManage debugvm <uuid|vmname> dumpvmcore [--filename=name]
```

Creates a system dump file of the specified VM. This file will have the standard ELF core format (with custom sections); see chapter 12.1.4, *VM core format*, page 249.

This corresponds to the `writecore` command in the debugger.

--filename=*filename*
: The name of the output file.

debugvm info

```
VBoxManage debugvm <uuid|vmname> info <item> [args...]
```

Displays info items relating to the VMM, device emulations and associated drivers.

This corresponds to the `info` command in the debugger.

info
: Name of the info item to display. The special name `help` will list all the available info items and hints about optional arguments.

args
: Optional argument string for the info item handler. Most info items does not take any extra arguments. Arguments not recognized are generally ignored.

debugvm injectnmi

```
VBoxManage debugvm <uuid|vmname> injectnmi
```

Causes a non-maskable interrupt (NMI) to be injected into the guest. This might be useful for certain debugging scenarios. What happens exactly is dependent on the guest operating system, but an NMI can crash the whole guest operating system. Do not use unless you know what you're doing.

debugvm log

```
VBoxManage debugvm <uuid|vmname> log [[--release] | [--debug]] [group-settings...]
```

Changes the group settings for either debug (`--debug`) or release (`--release`) logger of the VM process.

The *`group-settings`* are typically strings on the form `em.e.f.l`, `hm=~0` and `-em.f`. Basic wildcards are supported for group matching. The `all` group is an alias for all the groups.

Please do keep in mind that the group settings are applied as modifications to the current ones.

This corresponds to the `log` command in the debugger.

debugvm logdest

```
VBoxManage debugvm <uuid|vmname> logdest [[--release] | [--debug]] [destinations...]
```

Changes the destination settings for either debug (`--debug`) or release (`--release`) logger of the VM process. For details on the destination format, the best source is src/VBox/Runtime/common/log/log.cpp.

The *`destinations`* is one or more mnemonics, optionally prefixed by "no" to disable them. Some of them take values after a ":" or "=" separator. Multiple mnemonics can be separated by space or given as separate arguments on the command line.

List of available destination:

file[=*file*], nofile
: Specifies a log file. It no filename is given, one will be generated based on the current UTC time and VM process name and placed in the current directory of the VM process. Note that this will currently not have any effect if the log file has already been opened.

dir=*directory*, nodir
: Specifies the output directory for log files. Note that this will currently not have any effect if the log file has already been opened.

history=*count*, nohistory
: A non-zero value enables log historization, with the value specifying how many old log files to keep.

histsize=*bytes*
: The max size of a log file before it is historized. Default is infinite.

histtime=*seconds*
: The max age (in seconds) of a log file before it is historized. Default is infinite.

ringbuffer, noringbuffer
: Only log to the log buffer until an explicit flush (e.g. via an assertion) occurs. This is fast and saves diskspace.

stdout, nostdout
: Write the log content to standard output.

stdout, nostdout
: Write the log content to standard error.

debugger, nodebugger
: Write the log content to the debugger, if supported by the host OS.

com, nocom
: Writes logging to the COM port. This is only applicable for raw-mode and ring-0 logging.

user, nouser
Custom destination which has no meaning to VM processes..

This corresponds to the `logdest` command in the debugger.

debugvm logflags

```
VBoxManage debugvm <uuid|vmname> logflags [[--release] | [--debug]] [flags...]
```

Changes the flags on either debug (`--debug`) or release (`--release`) logger of the VM process. Please note that the modifications are applied onto the existing changes, they are not replacing them.

The *flags* are a list of flag mnemonics, optionally prefixed by a "no", "¡', "~" or "-" to negate their meaning. The "+" prefix can be used to undo previous negation or use as a separator, though better use whitespace or separate arguments for that.

List of log flag mnemonics, with their counter form where applicable (asterisk indicates defaults):

enabled*, disabled
Enables or disables logging.

buffered, unbuffered*
Enabling buffering of log output before it hits the destinations.

writethrough(/writethru)
Whether to open the destination file with writethru buffering settings or not.

flush
Enables flushing of the output file (to disk) after each log statement.

lockcnts
Prefix each log line with lock counts for the current thread.

cpuid
Prefix each log line with the ID of the current CPU.

pid
Prefix each log line with the current process ID.

flagno
Prefix each log line with the numberic flags corresponding to the log statement.

flag
Prefix each log line with the flag mnemonics corresponding to the log statement.

groupno
Prefix each log line with the log group number for the log statement producing it.

group
Prefix each log line with the log group name for the log statement producing it.

tid
Prefix each log line with the current thread identifier.

thread
Prefix each log line with the current thread name.

time
Prefix each log line with the current UTC wall time.

timeprog
Prefix each log line with the current monotonic time since the start of the program.

msprog
Prefix each log line with the current monotonic timestamp value in milliseconds since the start of the program.

ts
Prefix each log line with the current monotonic timestamp value in nanoseconds.

tsc
Prefix each log line with the current CPU timestamp counter (TSC) value.

rel, abs*
Selects the whether `ts` and `tsc` prefixes should be displayed as relative to the previous log line or as absolute time.

hex*, dec
Selects the whether the `ts` and `tsc` prefixes should be formatted as hexadecimal or decimal.

custom
Custom log prefix, has by default no meaning for VM processes.

usecrlf, uself*
Output with DOS style (CRLF) or just UNIX style (LF) line endings.

overwrite*, append
Overwrite the destination file or append to it.

This corresponds to the `logflags` command in the debugger.

debugvm osdetect

```
VBoxManage debugvm <uuid|vmname> osdetect
```

Make the VMM's debugger facility (re)-detect the guest operating system (OS). This will first load all debugger plug-ins.
This corresponds to the `detect` command in the debugger.

debugvm osinfo

```
VBoxManage debugvm <uuid|vmname> osinfo
```

Displays information about the guest operating system (OS) previously detected by the VMM's debugger facility.

debugvm osdmesg

```
VBoxManage debugvm <uuid|vmname> osdmesg [--lines=lines]
```

Displays the guest OS kernel log, if detected and supported.

--lines=*lines*
Number of lines of the log to display, counting from the end. The default is infinite.

debugvm getregisters

```
VBoxManage debugvm <uuid|vmname> getregisters [--cpu=id] [reg-set.reg-name...]
```

Retrieves register values for guest CPUs and emulated devices.

reg-set.reg-name
One of more registers, each having one of the following forms:

1. register-set.register-name.sub-field
2. register-set.register-name
3. cpu-register-name.sub-field
4. cpu-register-name
5. all

The *all* form will cause all registers to be shown (no sub-fields). The registers names are case-insensitive.

--cpu=*id*
Selects the CPU register set when specifying just a CPU register (3rd and 4th form). The default is 0.

debugvm setregisters

```
VBoxManage debugvm <uuid|vmname> setregisters [--cpu=id] [reg-set.reg-name=value...]
```

Changes register values for guest CPUs and emulated devices.

reg-set.reg-name=value
One of more register assignment, each having one of the following forms:

1. register-set.register-name.sub-field=value
2. register-set.register-name=value
3. cpu-register-name.sub-field=value
4. cpu-register-name=value

The value format should be in the same style as what `getregisters` displays, with the exception that both octal and decimal can be used instead of hexadecimal.

--cpu=*id*
Selects the CPU register set when specifying just a CPU register (3rd and 4th form). The default is 0.

debugvm show

```
VBoxManage debugvm <uuid|vmname> show [[--human-readable] | [--sh-export] |
    [--sh-eval] | [--cmd-set]] [settings-item...]
```

Shows logging settings for the VM.

--human-readable
Selects human readable output.

--sh-export
Selects output format as bourne shell style `export` commands.

--sh-eval
Selects output format as bourne shell style `eval` command input.

--cmd-set
Selects output format as DOS style `SET` commands.

settings-item
What to display. One or more of the following:

- logdbg-settings - debug log settings.
- logrel-settings - release log settings.
- log-settings - alias for both debug and release log settings.

debugvm stack

```
VBoxManage debugvm <uuid|vmname> stack [--cpu=id]
```

Unwinds the guest CPU stacks to the best of our ability. It is recommended to first run the `osdetect` command, as this gives both symbols and perhaps unwind information.

--cpu=*id*
Selects a single guest CPU to display the stack for. The default is all CPUs.

debugvm statistics

```
VBoxManage debugvm <uuid|vmname> statistics [--reset] [--descriptions] [--pattern=pattern]
```

Displays or resets VMM statistics.
Retrieves register values for guest CPUs and emulated devices.

--pattern=*pattern*
DOS/NT-style wildcards patterns for selecting statistics. Multiple patterns can be specified by using the '|' (pipe) character as separator.

--reset
Select reset instead of display mode.

8.41 VBoxManage extpack

Extension package management.

Synopsis

```
VBoxManage extpack install [--replace] <tarball>
VBoxManage extpack uninstall [--force] <name>
VBoxManage extpack cleanup
```

Description

extpack install

```
VBoxManage extpack install [--replace] <tarball>
```

Installs a new extension pack on the system. This command will fail if an older version of the same extension pack is already installed. The `--replace` option can be used to uninstall any old package before the new one is installed.

--replace
Uninstall existing extension pack version.

tarball
The file containing the extension pack to be installed.

extpack uninstall

```
VBoxManage extpack uninstall [--force] <name>
```

Uninstalls an extension pack from the system. The subcommand will also succeed in the case where the specified extension pack is not present on the system. You can use `VBoxManage list extpacks` to show the names of the extension packs which are currently installed.

--force
Overrides most refusals to uninstall an extension pack

name
The name of the extension pack to be uninstalled.

extpack cleanup

```
VBoxManage extpack cleanup
```

Used to remove temporary files and directories that may have been left behind if a previous install or uninstall command failed.

Examples

How to list extension packs:

```
$ VBoxManage list extpacks
Extension Packs: 1
Pack no. 0:   Oracle VM VirtualBox Extension Pack
Version:      4.1.12
Revision:     77218
Edition:
Description:  USB 2.0 Host Controller, VirtualBox RDP, PXE ROM with E1000 support.
VRDE Module:  VBoxVRDP
Usable:       true
Why unusable:
```

How to remove an extension pack:

```
$ VBoxManage extpack uninstall "Oracle VM VirtualBox Extension Pack"
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
Successfully uninstalled "Oracle VM VirtualBox Extension Pack".
```

9 Advanced topics

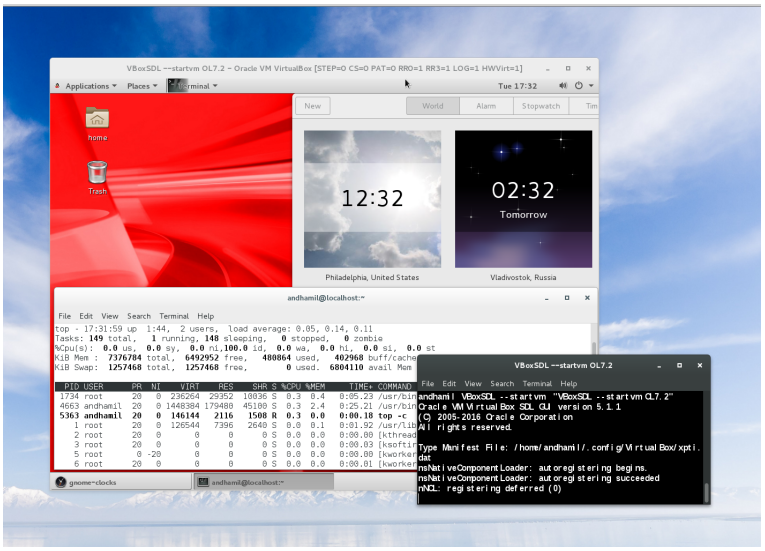
9.1 VBoxSDL, the simplified VM displayer

9.1.1 Introduction

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.

Note: VBoxSDL is not available on the Mac OS X host platform.

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 --startvm <vm>
```

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

9.1.2 Secure labeling with VBoxSDL

When running guest operating systems in full screen 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 RGB color values. The following syntax is used to enable secure labeling:

```
VBoxSDL --startvm "VM name"
      --securelabel --seclabelfnt ~/fonts/arial.ttf
      --seclabelsiz 14 --seclabelfgcol 00FF00 --seclabelbgcol 00FFFF
```

In addition to enabling secure labeling, a TrueType font has to be supplied. To use another font size than 12 point use the parameter `--seclabelsiz`.

The label text can be set with

```
VBoxManage setextradata "VM name" "VBoxSDL/SecureLabel" "The Label"
```

Changing this label will take effect immediately.

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.)

9.1.3 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
```

9.2 Automated guest logons

VirtualBox provides Guest Addition modules for Windows, Linux and Solaris to enable automated logons on the guest.

When a guest operating system is running in a virtual machine, it might be desirable to perform coordinated and automated logons 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.)

9.2.1 Automated Windows guest logons

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). With Windows Vista and Windows 7, the GINA modules were replaced with a new mechanism called "credential providers". The VirtualBox Guest Additions for Windows come with both, a GINA and a credential provider module, and therefore enable any Windows guest to perform automated logons.

To activate the VirtualBox GINA or credential provider module, install the Guest Additions with using the command line switch `/with_autologon`. All the following manual steps required for installing these modules will be then done by the installer.

To manually install the VirtualBox GINA module, extract the Guest Additions (see chapter 4.2.1.4, *Manual file extraction*, page 65) and copy the file `VBoxGINA.dll` 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 module is implemented as a wrapper around the standard Windows GINA module (`MSGINA.DLL`). As a result, it will most likely not work correctly with 3rd party GINA modules.

To manually install the VirtualBox credential provider module, extract the Guest Additions (see chapter 4.2.1.4, *Manual file extraction*, page 65) and copy the file `VBoxCredProv.dll` to the Windows `SYSTEM32` directory. Then, in the registry, create the following keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\
          Authentication\Credential Providers\{275D3BCC-22BB-4948-A7F6-3A3054EBA92B}

HKEY_CLASSES_ROOT\CLSID\{275D3BCC-22BB-4948-A7F6-3A3054EBA92B}

HKEY_CLASSES_ROOT\CLSID\{275D3BCC-22BB-4948-A7F6-3A3054EBA92B}\InprocServer32
```

with all default values (the key named `(Default)` in each key) set to `VBoxCredProv`. After that a new string named

```
HKEY_CLASSES_ROOT\CLSID\{275D3BCC-22BB-4948-A7F6-3A3054EBA92B}\InprocServer32\ThreadingModel
```

with a value of `Apartment` has to be created.

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 logon modules (GINA or credential provider) using the VirtualBox Guest Additions device driver. When Windows is in "logged out" mode, the logon modules will constantly poll for credentials and if they

are present, a logon will be attempted. After retrieving the credentials, the logon modules 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.

Depending on the particular variant of the Windows guest, the following restrictions apply:

1. For **Windows XP guests,** the logon subsystem needs to be configured to use the classic logon dialog as the VirtualBox GINA module does not support the XP-style welcome dialog.

2. For **Windows Vista, Windows 7 and Windows 8 guests,** the logon subsystem does not support the so-called Secure Attention Sequence (`CTRL+ALT+DEL`). As a result, the guest's group policy settings need to be changed to not use the Secure Attention Sequence. Also, the user name given is only compared to the true user name, not the user friendly name. This means that when you rename a user, you still have to supply the original user name (internally, Windows never renames user accounts).

3. Auto-logon handling of the built-in Windows Remote Desktop Service (formerly known as Terminal Services) is disabled by default. To enable it, create the registry key

   `HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\VirtualBox Guest Additions\AutoLogon`

   with a `DWORD` value of `1`.

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.

9.2.2 Automated Linux/Unix guest logons

Starting with version 3.2, VirtualBox provides a custom PAM module (Pluggable Authentication Module) which can be used to perform automated guest logons on platforms which support this framework. Virtually all modern Linux/Unix distributions rely on PAM.

For automated logons on Ubuntu (or Ubuntu-derived) distributions using LightDM as the display manager, please see chapter 9.2.2.1, *VirtualBox Greeter for Ubuntu / LightDM*, page 190.

The `pam_vbox.so` module itself **does not** do an actual verification of the credentials passed to the guest OS; instead it relies on other modules such as `pam_unix.so` or `pam_unix2.so` down in the PAM stack to do the actual validation using the credentials retrieved by `pam_vbox.so`. Therefore `pam_vbox.so` has to be on top of the authentication PAM service list.

> **Note:** The `pam_vbox.so` only supports the `auth` primitive. Other primitives such as `account`, `session` or `password` are not supported.

The `pam_vbox.so` module is shipped as part of the Guest Additions but it is not installed and/or activated on the guest OS by default. In order to install it, it has to be copied from `/opt/VBoxGuestAdditions-<version>/lib/VBoxGuestAdditions/` to the security modules directory, usually `/lib/security/` on 32-bit guest Linuxes or `/lib64/security/` on 64-bit ones. Please refer to your guest OS documentation for the correct PAM module directory.

For example, to use `pam_vbox.so` with a Ubuntu Linux guest OS and GDM (the GNOME Desktop Manager) to logon users automatically with the credentials passed by the host, the guest OS has to be configured like the following:

1. The `pam_vbox.so` module has to be copied to the security modules directory, in this case it is `/lib/security`.

2. Edit the PAM configuration file for GDM found at `/etc/pam.d/gdm`, adding the line `auth requisite pam_vbox.so` at the top. Additionally, in most Linux distributions there is a file called `/etc/pam.d/common-auth`. This file is included in many other services (like the GDM file mentioned above). There you also have to add the line `auth requisite pam_vbox.so`.

3. If authentication against the shadow database using `pam_unix.so` or `pam_unix2.so` is desired, the argument `try_first_pass` for `pam_unix.so` or `use_first_pass` for `pam_unix2.so` is needed in order to pass the credentials from the VirtualBox module to the shadow database authentication module. For Ubuntu, this needs to be added to `/etc/pam.d/common-auth`, to the end of the line referencing `pam_unix.so`. This argument tells the PAM module to use credentials already present in the stack, i.e. the ones provided by the VirtualBox PAM module.

Warning: An incorrectly configured PAM stack can effectively prevent you from logging into your guest system!

To make deployment easier, you can pass the argument `debug` right after the `pam_vbox.so` statement. Debug log output will then be recorded using syslog.

Note: By default, pam_vbox will not wait for credentials to arrive from the host, in other words: When a login prompt is shown (for example by GDM/KDM or the text console) and pam_vbox does not yet have credentials it does not wait until they arrive. Instead the next module in the PAM stack (depending on the PAM configuration) will have the chance for authentication.

Starting with VirtualBox 4.1.4 pam_vbox supports various guest property parameters which all reside in `/VirtualBox/GuestAdd/PAM/`. These parameters allow pam_vbox to wait for credentials to be provided by the host and optionally can show a message while waiting for those. The following guest properties can be set:

1. `CredsWait`: Set to "1" if pam_vbox should start waiting until credentials arrive from the host. Until then no other authentication methods such as manually logging in will be available. If this property is empty or get deleted no waiting for credentials will be performed and pam_vbox will act like before (see paragraph above). This property must be set read-only for the guest (`RDONLYGUEST`).

2. `CredsWaitAbort`: Aborts waiting for credentials when set to any value. Can be set from host and the guest.

3. `CredsWaitTimeout`: Timeout (in seconds) to let pam_vbox wait for credentials to arrive. When no credentials arrive within this timeout, authentication of pam_vbox will be set to failed and the next PAM module in chain will be asked. If this property is not specified, set to "0" or an invalid value, an infinite timeout will be used. This property must be set read-only for the guest (`RDONLYGUEST`).

To customize pam_vbox further there are the following guest properties:

1. `CredsMsgWaiting`: Custom message showed while pam_vbox is waiting for credentials from the host. This property must be set read-only for the guest (`RDONLYGUEST`).

2. `CredsMsgWaitTimeout`: Custom message showed when waiting for credentials by pam_vbox timed out, e.g. did not arrive within time. This property must be set read-only for the guest (`RDONLYGUEST`).

> **Note:** If a pam_vbox guest property does not have set the right flags (`RDONLYGUEST`) this property will be ignored then and - depending on the property - a default value will be set. This can result in pam_vbox not waiting for credentials. Consult the appropriate syslog file for more information and use the `debug` option.

9.2.2.1 VirtualBox Greeter for Ubuntu / LightDM

Starting with version 4.2.12, VirtualBox comes with an own greeter module named vbox-greeter which can be used with LightDM 1.0.1 or later. LightDM is the default display manager since Ubuntu 10.11 and therefore also can be used for automated guest logons.

vbox-greeter does not need the pam_vbox module described above in order to function – it comes with its own authentication mechanism provided by LightDM. However, to provide maximum of flexibility both modules can be used together on the same guest.

As for the pam_vbox module, vbox-greeter is shipped as part of the Guest Additions but it is not installed and/or activated on the guest OS by default For installing vbox-greeter automatically upon Guest Additions installation, use the `--with-autologon` switch when starting the VBoxLinuxAdditions.run file:

```
# ./VBoxLinuxAdditions.run -- --with-autologon
```

For manual or postponed installation, the `vbox-greeter.desktop` file has to be copied from `/opt/VBoxGuestAdditions-<version>/shared/VBoxGuestAdditions/` to the `xgreeters` directory, usually `/usr/share/xgreeters/`. Please refer to your guest OS documentation for the correct LightDM greeter directory.

The vbox-greeter module itself already was installed by the VirtualBox Guest Additions installer and resides in `/usr/sbin/`. To enable vbox-greeter as the standard greeter module, the file `/etc/lightdm/lightdm.conf` needs to be edited:

```
[SeatDefaults]
greeter-session=vbox-greeter
```

> **Note:** The LightDM server needs to be fully restarted in order to get vbox-greeter used as the default greeter. As root, do a `service lightdm --full-restart` on Ubuntu, or simply restart the guest.

> **Note:** vbox-greeter is independent of the graphical session chosen by the user (like Gnome, KDE, Unity etc). However, it requires FLTK 1.3 for representing its own user interface.

There are numerous guest properties which can be used to further customize the login experience. For automatically logging in users, the same guest properties apply as for pam_vbox, see chapter 9.2.2, *Automated Linux/Unix guest logons*, page 188.

In addition to the above mentioned guest properties, vbox-greeter allows further customization of its user interface. These special guest properties all reside in `/VirtualBox/GuestAdd/Greeter/`:

1. `HideRestart`: Set to "1" if vbox-greeter should hide the button to restart the guest. This property must be set read-only for the guest (`RDONLYGUEST`).
2. `HideShutdown`: Set to "1" if vbox-greeter should hide the button to shutdown the guest. This property must be set read-only for the guest (`RDONLYGUEST`).
3. `BannerPath`: Path to a .PNG file for using it as a banner on the top. The image size must be 460 x 90 pixels, any bit depth. This property must be set read-only for the guest (`RDONLYGUEST`).
4. `UseTheming`: Set to "1" for turning on the following theming options. This property must be set read-only for the guest (`RDONLYGUEST`).
5. `Theme/BackgroundColor`: Hexadecimal RRGGBB color for the background. This property must be set read-only for the guest (`RDONLYGUEST`).
6. `Theme/LogonDialog/HeaderColor`: Hexadecimal RRGGBB foreground color for the header text. This property must be set read-only for the guest (`RDONLYGUEST`).
7. `Theme/LogonDialog/BackgroundColor`: Hexadecimal RRGGBB color for the logon dialog background. This property must be set read-only for the guest (`RDONLYGUEST`).
8. `Theme/LogonDialog/ButtonColor`: Hexadecimal RRGGBB background color for the logon dialog button. This property must be set read-only for the guest (`RDONLYGUEST`).

Note: The same restrictions for the guest properties above apply as for the ones specified in the pam_vbox section.

9.3 Advanced configuration for Windows guests

9.3.1 Automated Windows system preparation

Beginning with Windows NT 4.0, Microsoft offers a "system preparation" tool (in short: Sysprep) to prepare a Windows system for deployment or redistribution. Whereas Windows 2000 and XP ship with Sysprep on the installation medium, the tool also is available for download on the Microsoft web site. In a standard installation of Windows Vista and 7, Sysprep is already included. Sysprep mainly consists of an executable called `sysprep.exe` which is invoked by the user to put the Windows installation into preparation mode.

Starting with VirtualBox 3.2.2, the Guest Additions offer a way to launch a system preparation on the guest operating system in an automated way, controlled from the host system. To achieve that, see chapter 4.8, *Guest control*, page 76 for using the feature with the special identifier `sysprep` as the program to execute, along with the user name `sysprep` and password `sysprep` for the credentials. Sysprep then gets launched with the required system rights.

Note: Specifying the location of "sysprep.exe" is **not possible** – instead the following paths are used (based on the operating system):

- `C:\sysprep\sysprep.exe` for Windows NT 4.0, 2000 and XP
- `%WINDIR%\System32\Sysprep\sysprep.exe` for Windows Vista, 2008 Server and 7

The Guest Additions will automatically use the appropriate path to execute the system preparation tool.

9.4 Advanced configuration for Linux and Solaris guests

9.4.1 Manual setup of selected guest services on Linux

The VirtualBox Guest Additions contain several different drivers. If for any reason you do not wish to set them all up, you can install the Guest Additions using the following command:

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

After this, you will need to at least compile the kernel modules by running the command

```
rcvboxadd setup
```

as root (you will need to replace *lib* by *lib64* on some 64bit guests), and on older guests without the udev service you will need to add the *vboxadd* service to the default runlevel to ensure that the modules get loaded.

To setup the time synchronization service, add the service vboxadd-service to the default runlevel. To set up the X11 and OpenGL part of the Guest Additions, run the command

```
rcvboxadd-x11 setup
```

(you do not need to enable any services for this).

To recompile the guest kernel modules, use this command:

```
rcvboxadd setup
```

After compilation you should reboot your guest to ensure that the new modules are actually used.

9.4.2 Guest graphics and mouse driver setup in depth

This section assumes that you are familiar with configuring the X.Org server using xorg.conf and optionally the newer mechanisms using hal or udev and xorg.conf.d. If not you can learn about them by studying the documentation which comes with X.Org.

The VirtualBox Guest Additions come with drivers for X.Org versions

- X11R6.8/X11R6.9 and XFree86 version 4.3 (vboxvideo_drv_68.o and vboxmouse_drv_68.o)
- X11R7.0 (vboxvideo_drv_70.so and vboxmouse_drv_70.so)
- X11R7.1 (vboxvideo_drv_71.so and vboxmouse_drv_71.so)
- X.Org Server versions 1.3 and later (vboxvideo_drv_13.so and vboxmouse_drv_13.so and so on).

By default these drivers can be found in the directory

`/opt/VBoxGuestAdditions-<version>/lib/VBoxGuestAdditions`

and the correct versions for the X server are symbolically linked into the X.Org driver directories.

For graphics integration to work correctly, the X server must load the vboxvideo driver (many recent X server versions look for it automatically if they see that they are running in VirtualBox) and for an optimal user experience the guest kernel drivers must be loaded and the Guest Additions tool VBoxClient must be running as a client in the X session. For mouse integration to work correctly, the guest kernel drivers must be loaded and in addition, in X servers from X.Org X11R6.8 to X11R7.1 and in XFree86 version 4.3 the right vboxmouse driver must be loaded and associated with /dev/mouse or /dev/psaux; in X.Org server 1.3 or later a driver for a PS/2 mouse must be loaded and the right vboxmouse driver must be associated with /dev/vboxguest.

The VirtualBox guest graphics driver can use any graphics configuration for which the virtual resolution fits into the virtual video memory allocated to the virtual machine (minus a small amount used by the guest driver) as described in chapter 3.5, *Display settings*, page 51. The driver will offer a range of standard modes at least up to the default guest resolution for all active guest monitors. In X.Org Server 1.3 and later the default mode can be changed by setting the output property VBOX_MODE to "<width>x<height>" for any guest monitor. When VBoxClient and the kernel drivers are active this is done automatically when the host requests a mode change. The driver for older versions can only receive new modes by querying the host for requests at regular intervals.

With pre-1.3 X Servers 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
```

9.5 CPU hot-plugging

With virtual machines running modern server operating systems, VirtualBox supports CPU hot-plugging.[1] Whereas on a physical computer this would mean that a CPU can be added or removed while the machine is running, VirtualBox supports adding and removing virtual CPUs while a virtual machine is running.

CPU hot-plugging works only with guest operating systems that support it. So far this applies only to Linux and Windows Server 2008 x64 Data Center Edition. Windows supports only hot-add while Linux supports hot-add and hot-remove but to use this feature with more than 8 CPUs a 64bit Linux guest is required.

At this time, CPU hot-plugging requires using the VBoxManage command-line interface. First, hot-plugging needs to be enabled for a virtual machine:

```
VBoxManage modifyvm "VM name" --cpuhotplug on
```

After that, the `--cpus` option specifies the maximum number of CPUs that the virtual machine can have:

```
VBoxManage modifyvm "VM name" --cpus 8
```

When the VM is off, you can then add and remove virtual CPUs with the `modifyvm --plugcpu` and `--unplugcpu` subcommands, which take the number of the virtual CPU as a parameter, like this:

```
VBoxManage modifyvm "VM name" --plugcpu 3
VBoxManage modifyvm "VM name" --unplugcpu 3
```

Note that CPU 0 can never be removed.

While the VM is running, CPUs can be added and removed with the `controlvm plugcpu` and `unplugcpu` commands instead:

```
VBoxManage controlvm "VM name" plugcpu 3
VBoxManage controlvm "VM name" unplugcpu 3
```

[1] Support for CPU hot-plugging was introduced with VirtualBox 3.2.

See chapter 8.8, *VBoxManage modifyvm*, page 128 and chapter 8.13, *VBoxManage controlvm*, page 143 for details.

With Linux guests, the following applies: To prevent ejection while the CPU is still used it has to be ejected from within the guest before. The Linux Guest Additions contain a service which receives hot-remove events and ejects the CPU. Also, after a CPU is added to the VM it is not automatically used by Linux. The Linux Guest Additions service will take care of that if installed. If not a CPU can be started with the following command:

```
echo 1 > /sys/devices/system/cpu/cpu<id>/online
```

9.6 PCI passthrough

When running on Linux hosts, with a recent enough kernel (at least version `2.6.31`) experimental host PCI devices passthrough is available.[2]

> **Note:** The PCI passthrough module is shipped as a VirtualBox extension package, which must be installed separately. See chapter 1.5, *Installing VirtualBox and extension packs*, page 16 for more information.

Essentially this feature allows to directly use physical PCI devices on the host by the guest even if host doesn't have drivers for this particular device. Both, regular PCI and some PCI Express cards, are supported. AGP and certain PCI Express cards are not supported at the moment if they rely on GART (Graphics Address Remapping Table) unit programming for texture management as it does rather non-trivial operations with pages remapping interfering with IOMMU. This limitation may be lifted in future releases.

To be fully functional, PCI passthrough support in VirtualBox depends upon an IOMMU hardware unit which is not yet too widely available. If the device uses bus mastering (i.e. it performs DMA to the OS memory on its own), then an IOMMU is required, otherwise such DMA transactions may write to the wrong physical memory address as the device DMA engine is programmed using a device-specific protocol to perform memory transactions. The IOMMU functions as translation unit mapping physical memory access requests from the device using knowledge of the guest physical address to host physical addresses translation rules.

Intel's solution for IOMMU is marketed as "Intel Virtualization Technology for Directed I/O" (VT-d), and AMD's one is called AMD-Vi. So please check if your motherboard datasheet has appropriate technology. Even if your hardware doesn't have a IOMMU, certain PCI cards may work (such as serial PCI adapters), but the guest will show a warning on boot and the VM execution will terminate if the guest driver will attempt to enable card bus mastering.

It is very common that the BIOS or the host OS disables the IOMMU by default. So before any attempt to use it please make sure that

1. Your motherboard has an IOMMU unit.
2. Your CPU supports the IOMMU.
3. The IOMMU is enabled in the BIOS.
4. The VM must run with VT-x/AMD-V and nested paging enabled.
5. Your Linux kernel was compiled with IOMMU support (including DMA remapping, see `CONFIG_DMAR` kernel compilation option). The PCI stub driver (`CONFIG_PCI_STUB`) is required as well.

[2] Experimental support for PCI passthrough was introduced with VirtualBox 4.1.

6. Your Linux kernel recognizes and uses the IOMMU unit (`intel_iommu=on` boot option could be needed). Search for DMAR and PCI-DMA in kernel boot log.

Once you made sure that the host kernel supports the IOMMU, the next step is to select the PCI card and attach it to the guest. To figure out the list of available PCI devices, use the `lspci` command. The output will look like this:

```
01:00.0 VGA compatible controller: ATI Technologies Inc Cedar PRO [Radeon HD 5450]
01:00.1 Audio device: ATI Technologies Inc Manhattan HDMI Audio [Mobility Radeon HD 5000 Series]
02:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168B PCI Express Gigabit
        Ethernet controller (rev 03)
03:00.0 SATA controller: JMicron Technology Corp. JMB362/JMB363 Serial ATA Controller (rev 03)
03:00.1 IDE interface: JMicron Technology Corp. JMB362/JMB363 Serial ATA Controller (rev 03)
06:00.0 VGA compatible controller: nVidia Corporation G86 [GeForce 8500 GT] (rev a1)
```

The first column is a PCI address (in format `bus:device.function`). This address could be used to identify the device for further operations. For example, to attach a PCI network controller on the system listed above to the second PCI bus in the guest, as device 5, function 0, use the following command:

```
VBoxManage modifyvm "VM name" --pciattach 02:00.0@01:05.0
```

To detach same device, use

```
VBoxManage modifyvm "VM name" --pcidetach 02:00.0
```

Please note that both host and guest could freely assign a different PCI address to the card attached during runtime, so those addresses only apply to the address of the card at the moment of attachment (host), and during BIOS PCI init (guest).

If the virtual machine has a PCI device attached, certain limitations apply:

1. Only PCI cards with non-shared interrupts (such as using MSI on host) are supported at the moment.
2. No guest state can be reliably saved/restored (as the internal state of the PCI card could not be retrieved).
3. Teleportation (live migration) doesn't work (for the same reason).
4. No lazy physical memory allocation. The host will preallocate the whole RAM required for the VM on startup (as we cannot catch physical hardware accesses to the physical memory).

9.7 Webcam passthrough

9.7.1 Using a host webcam in the guest

VirtualBox 4.3 includes an experimental feature which allows a guest to use a host webcam. This complements the general USB passthrough support which was the typical way of using host webcams in earlier versions. The webcam passthrough support can handle non-USB video sources in theory, but this is completely untested.

> **Note:** The webcam passthrough module is shipped as part of the Oracle VM VirtualBox extension pack, which must be installed separately. See chapter 1.5, *Installing VirtualBox and extension packs*, page 16 for more information.

The host webcam can be attached to the VM using "Devices" menu in the VM menu bar. The "Webcams" menu contains a list of available video input devices on the host. Clicking on a webcam name attaches or detaches the corresponding host device.

The VBoxManage command line tool can be used to enable webcam passthrough. Please see the host-specific sections below for additional details. The following commands are available:

- Get a list of host webcams (or other video input devices):

  ```
  VBoxManage list webcams
  ```

  The output format:

  ```
  alias "user friendly name"
  host path or identifier
  ```

  The alias can be used as a shortcut in other commands. Alias '.0' means default video input device on the host, '.1', '.2', etc mean first, second, etc video input device. The device order is host-specific.

- Attach a webcam to a running VM:

  ```
  VBoxManage controlvm "VM name" webcam attach [host_path|alias [settings]]
  ```

  This will attach a USB webcam device to the guest.

  The `settings` parameter is a string `Setting1=Value1;Setting2=Value2`, which allows to configure the emulated webcam device. The following settings are supported:

  - `MaxFramerate` The highest rate at which video frames are sent to the guest. A higher frame rate requires more CPU power. Therefore sometimes it is useful to set a lower limit. Default is no limit and allow the guest to use all frame rates supported by the host webcam.
  - `MaxPayloadTransferSize` How many bytes the emulated webcam can send to the guest at a time. Default value is 3060 bytes, which is used by some webcams. Higher values can slightly reduce CPU load, if the guest is able to use larger buffers. However, a high `MaxPayloadTransferSize` might be not supported by some guests.

- Detach a webcam from a running VM:

  ```
  VBoxManage controlvm "VM name" webcam detach [host_path|alias]
  ```

- List webcams attached to a running VM:

  ```
  VBoxManage controlvm "VM name" webcam list
  ```

  The output contains path or alias which was used in 'webcam attach' command for each attached webcam.

9.7.2 Windows hosts

When the webcam device is detached from the host, the emulated webcam device is automatically detached from the guest.

9.7.3 Mac OS X hosts

OS X version 10.9 or newer is required.

When the webcam device is detached from the host, the emulated webcam device remains attached to the guest and must be manually detached using the `VBoxManage controlvm "VM name" webcam detach ...` command.

9.7.4 Linux and Solaris hosts

When the webcam is detached from the host the emulated webcam device is automatically detached from the guest only if the webcam is streaming video. If the emulated webcam is inactive it should be manually detached using the `VBoxManage controlvm "VM name" webcam detach ...` command.

Aliases `.0` and `.1` are mapped to `/dev/video0`, alias `.2` is mapped to `/dev/video1` and so forth.

9.8 Advanced display configuration

9.8.1 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 "VM name" "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.

9.8.2 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 full screen 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.

9.9 Advanced storage configuration

9.9.1 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 79), 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 without going through the host OS file system. The actual performance difference for image files vs. raw disk varies greatly depending on the overhead of the host file system, whether dynamically growing images are used, and on host OS caching strategies. The caching indirectly also affects other aspects such as failure behavior, i.e. whether the virtual disk contains all data written before a host OS crash. Consult your host OS documentation for details on this.

> **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. 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 VirtualBox Manager (chapter 5.3, *The Virtual Media Manager*, page 83) or `VBoxManage` to assign the image to a virtual machine.

9.9.1.1 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`. On a Mac OS X host, instead of the above device specification use e.g. `/dev/disk1`. Note that on OS X you can only get access to an entire disk if no volume is mounted from it.

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. On some host platforms (e.g. Windows Vista and later), raw disk access may be restricted and not permitted by the host OS in some situations.

Just like with regular disk images, this does not automatically attach the newly created image to a virtual machine. This can be done with e.g.

```
VBoxManage storageattach WindowsXP --storagectl "IDE Controller"
      --port 0 --device 0 --type hdd --medium /path/to/file.vmdk
```

When this is done the selected virtual machine will boot from the specified physical disk.

9.9.1.2 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`. On a Mac OS X host, instead of the above device specification use e.g. `/dev/disk1`. Note that on OS X you can only use partitions which are not mounted (eject the respective volume first). Partition numbers are the same on Linux, Windows and Mac OS X 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.

The created image can be attached to a storage controller in a VM configuration as usual.

9.9.2 Configuring the hard disk vendor product data (VPD)

VirtualBox reports vendor product data for its virtual hard disks which consist of hard disk serial number, firmware revision and model number. These can be changed using the following commands:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/SerialNumber" "serial"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/FirmwareRevision" "firmware"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/ModelNumber" "model"
```

The serial number is a 20 byte alphanumeric string, the firmware revision an 8 byte alphanumeric string and the model number a 40 byte alphanumeric string. Instead of "Port0" (referring to the first port), specify the desired SATA hard disk port.

The above commands apply to virtual machines with an AHCI (SATA) controller. The commands for virtual machines with an IDE controller are:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/piix3ide/0/Config/PrimaryMaster/SerialNumber" "serial"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/piix3ide/0/Config/PrimaryMaster/FirmwareRevision" "firmware"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/piix3ide/0/Config/PrimaryMaster/ModelNumber" "model"
```

For hard disks it's also possible to mark the drive as having a non-rotational medium with:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/NonRotational" "1"
```

Additional three parameters are needed for CD/DVD drives to report the vendor product data:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/ATAPIVendorId" "vendor"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/ATAPIProductId" "product"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/Config/Port0/ATAPIRevision" "revision"
```

The vendor id is an 8 byte alphanumeric string, the product id an 16 byte alphanumeric string and the revision a 4 byte alphanumeric string. Instead of "Port0" (referring to the first port), specify the desired SATA hard disk port.

9.9.3 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. Please see chapter 5.10, *iSCSI servers*, page 91; chapter 6.6, *Internal networking*, page 98; and chapter 8.18, *VBoxManage storageattach*, page 150 for additional information.

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 eight commands must first be issued:

```
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/Trusted 1
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/Config/MAC 08:00:27:01:02:0f
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/Config/IP 10.0.9.1
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/Config/Netmask 255.255.255.0
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/LUN#0/Driver IntNet
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/LUN#0/Config/Network MyIntNet
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/LUN#0/Config/TrunkType 2
VBoxManage setextradata "VM name" VBoxInternal/Devices/IntNetIP/0/LUN#0/Config/IsService 1
```

Finally the iSCSI disk must be attached with the `--intnet` option to tell the iSCSI initiator to use internal networking:

```
VBoxManage storageattach ... --medium iscsi
         --server 10.0.9.30 --target iqn.2008-12.com.sun:sampletarget --intnet
```

Compared to a "regular" iSCSI setup, IP address of the target *must* be specified as a numeric IP address, as there is no DNS resolver for internal networking.

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.

9.10 Legacy commands for 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 3.9, *Serial ports*, page 55.

> **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 "VM name"
      "VBoxInternal/Devices/serial/0/Config/IRQ" 4
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/serial/0/Config/IOBase" 0x3f8
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/serial/0/LUN#0/Driver" Char
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/serial/0/LUN#0/AttachedDriver/Driver" NamedPipe
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/serial/0/LUN#0/AttachedDriver/Config/Location" "\\.\pipe\vboxCOM1"
VBoxManage setextradata "VM name"
      "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 configuration 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.

9.11 Fine-tuning the VirtualBox NAT engine

9.11.1 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. So `x` is 2 when there is only one

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 "VM name" --natnet1 "192.168/16"
```

This command would reserve the network addresses from `192.168.0.0` to `192.168.254.254` for the first NAT network instance of "VM name". The guest IP would be assigned to `192.168.0.15` and the default gateway could be found at `192.168.0.2`.

9.11.2 Configuring the boot server (next server) of a NAT network interface

For network booting in NAT mode, by default VirtualBox uses a built-in TFTP server at the IP address 10.0.2.4. This default behavior should work fine for typical remote-booting scenarios. However, it is possible to change the boot server IP and the location of the boot image with the following commands:

```
VBoxManage modifyvm "VM name" --nattftpserver1 10.0.2.2
VBoxManage modifyvm "VM name" --nattftpfile1 /srv/tftp/boot/MyPXEBoot.pxe
```

9.11.3 Tuning TCP/IP buffers for NAT

The VirtualBox NAT stack performance is often determined by its interaction with the host's TCP/IP stack and the size of several buffers (`SO_RCVBUF` and `SO_SNDBUF`). For certain setups users might want to adjust the buffer size for a better performance. This can by achieved using the following commands (values are in kilobytes and can range from 8 to 1024):

```
VBoxManage modifyvm "VM name" --natsettings1 16000,128,128,0,0
```

This example illustrates tuning the NAT settings. The first parameter is the MTU, then the size of the socket's send buffer and the size of the socket's receive buffer, the initial size of the TCP send window, and lastly the initial size of the TCP receive window. Note that specifying zero means fallback to the default value.

Each of these buffers has a default size of 64KB and default MTU is 1500.

9.11.4 Binding NAT sockets to a specific interface

By default, VirtualBox's NAT engine will route TCP/IP packets through the default interface assigned by the host's TCP/IP stack. (The technical reason for this is that the NAT engine uses sockets for communication.) If, for some reason, you want to change this behavior, you can tell the NAT engine to bind to a particular IP address instead. Use the following command:

```
VBoxManage modifyvm "VM name" --natbindip1 "10.45.0.2"
```

After this, all outgoing traffic will be sent through the interface with the IP address 10.45.0.2. Please make sure that this interface is up and running prior to this assignment.

9.11.5 Enabling DNS proxy in NAT mode

The NAT engine by default offers the same DNS servers to the guest that are configured on the host. In some scenarios, it can be desirable to hide the DNS server IPs from the guest, for example when this information can change on the host due to expiring DHCP leases. In this case, you can tell the NAT engine to act as DNS proxy using the following command:

```
VBoxManage modifyvm "VM name" --natdnsproxy1 on
```

9.11.6 Using the host's resolver as a DNS proxy in NAT mode

For resolving network names, the DHCP server of the NAT engine offers a list of registered DNS servers of the host. If for some reason you need to hide this DNS server list and use the host's resolver settings, thereby forcing the VirtualBox NAT engine to intercept DNS requests and forward them to host's resolver, use the following command:

```
VBoxManage modifyvm "VM name" --natdnshostresolver1 on
```

Note that this setting is similar to the DNS proxy mode, however whereas the proxy mode just forwards DNS requests to the appropriate servers, the resolver mode will interpret the DNS requests and use the host's DNS API to query the information and return it to the guest.

9.11.6.1 User-defined host name resolving

In some cases it might be useful to intercept the name resolving mechanism, providing a user-defined IP address on a particular DNS request. The intercepting mechanism allows the user to map not only a single host but domains and even more complex naming conventions if required.

The following command sets a rule for mapping a name to a specified IP:

```
VBoxManage setextradata "VM name" \
      "VBoxInternal/Devices/{pcnet,e1000}/0/LUN#0/AttachedDriver/Config/HostResolverMappings/ \
      <unique rule name of interception rule>/HostIP" <IPv4>
VBoxManage setextradata "VM name" \
      "VBoxInternal/Devices/{pcnet,e1000}/0/LUN#0/AttachedDriver/Config/HostResolverMappings/ \
      <unique rule name>/HostName" <name of host>
```

The following command sets a rule for mapping a pattern name to a specified IP:

```
VBoxManage setextradata "VM name" \
      "VBoxInternal/Devices/{pcnet,e1000}/0/LUN#0/AttachedDriver/Config/HostResolverMappings/ \
      <unique rule name>/HostIP" <IPv4>
VBoxManage setextradata "VM name" \
      "VBoxInternal/Devices/{pcnet,e1000}/0/LUN#0/AttachedDriver/Config/HostResolverMappings/ \
      <unique rule name>/HostNamePattern" <hostpattern>
```

The host pattern may include `"|", "?" and "*"`.

This example demonstrates how to instruct the host-resolver mechanism to resolve all domain and probably some mirrors of www.blocked-site.info site with IP 127.0.0.1:

```
VBoxManage setextradata "VM name" \
      "VBoxInternal/Devices/e1000/0/LUN#0/AttachedDriver/Config/HostResolverMappings/ \
      all_blocked_site/HostIP" 127.0.0.1
VBoxManage setextradata "VM name" \
      "VBoxInternal/Devices/e1000/0/LUN#0/AttachedDriver/Config/HostResolverMappings/ \
      all_blocked_site/HostNamePattern" "*.blocked-site.*|*.fb.org"
```

The host resolver mechanism should be enabled to use user-defined mapping rules, otherwise they don't have any effect.

9.11.7 Configuring aliasing of the NAT engine

By default, the NAT core uses aliasing and uses random ports when generating an alias for a connection. This works well for the most protocols like SSH, FTP and so on. Though some protocols might need a more transparent behavior or may depend on the real port number the packet was sent from. It is possible to change the NAT mode via the VBoxManage frontend with the following commands:

```
VBoxManage modifyvm "VM name" --nataliasmode1 proxyonly
```

and

```
VBoxManage modifyvm "Linux Guest" --nataliasmode1 sameports
```

The first example disables aliasing and switches NAT into transparent mode, the second example enforces preserving of port values. These modes can be combined if necessary.

9.12 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. In case your VM is configured to use EFI firmware you need to replace `pcbios` by `efi` in the keys.

DMI BIOS information (type 0)

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVendor"        "BIOS Vendor"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVersion"       "BIOS Version"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseDate"   "BIOS Release Date"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMajor"  1
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMinor"  2
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMajor" 3
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMinor" 4
```

DMI system information (type 1)

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemVendor"      "System Vendor"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemProduct"     "System Product"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemVersion"     "System Version"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemSerial"      "System Serial"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemSKU"         "System SKU"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemFamily"      "System Family"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemUuid"
                                                "9852bf98-b83c-49db-a8de-182c42c7226b"
```

DMI board information (type 2)

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardVendor"       "Board Vendor"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardProduct"      "Board Product"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardVersion"      "Board Version"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardSerial"       "Board Serial"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardAssetTag"     "Board Tag"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardLocInChass"   "Board Location"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiBoardBoardType"    10
```

DMI system enclosure or chassis (type 3)

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiChassisVendor"     "Chassis Vendor"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiChassisType"       3
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiChassisVersion"    "Chassis Version"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiChassisSerial"     "Chassis Serial"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiChassisAssetTag"   "Chassis Tag"
```

DMI processor information (type 4)

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiProcManufacturer"  "GenuineIntel"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiProcVersion"       "Pentium(R) III"
```

DMI OEM strings (type 11)

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiOEMVBoxVer"        "vboxVer_1.2.3"
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiOEMVBoxRev"        "vboxRev_12345"
```

If a DMI string is not set, the default value of VirtualBox is used. To set an empty string use `"<EMPTY>"`.

Note that in the above list, all quoted parameters (DmiBIOSVendor, DmiBIOSVersion but not DmiBIOSReleaseMajor) are expected to be strings. If such a string is a valid number, the parameter is treated as number and the VM will most probably refuse to start with an `VERR_CFGM_NOT_STRING` error. In that case, use `"string:<value>"`, for instance

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/pcbios/0/Config/DmiSystemSerial"      "string:1234"
```

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
```

9.13 Configuring the custom ACPI table

VirtualBox can be configured to present an custom ACPI table to the guest. Use the following command to configure this:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/acpi/0/Config/CustomTable" "/path/to/table.bin"
```

Configuring a custom ACPI table can prevent Windows Vista and Windows 7 from asking for a new product key. On Linux hosts, one of the host tables can be read from /sys/firmware/acpi/tables/.

9.14 Fine-tuning timers and time synchronization

9.14.1 Configuring the guest time stamp counter (TSC) to reflect guest execution

By default, VirtualBox keeps all sources of time visible to the guest synchronized to a single time source, the monotonic host time. This reflects the assumptions of many guest operating systems, which expect all time sources to reflect "wall clock" time. In special circumstances it may be useful however to make the TSC (time stamp counter) in the guest reflect the time actually spent executing the guest.

This special TSC handling mode can be enabled on a per-VM basis, and for best results must be used only in combination with hardware virtualization. To enable this mode use the following command:

```
VBoxManage setextradata "VM name" "VBoxInternal/TM/TSCTiedToExecution" 1
```

To revert to the default TSC handling mode use:

```
VBoxManage setextradata "VM name" "VBoxInternal/TM/TSCTiedToExecution"
```

Note that if you use the special TSC handling mode with a guest operating system which is very strict about the consistency of time sources you may get a warning or error message about the timing inconsistency. It may also cause clocks to become unreliable with some guest operating systems depending on how they use the TSC.

9.14.2 Accelerate or slow down the guest clock

For certain purposes it can be useful to accelerate or to slow down the (virtual) guest clock. This can be achieved as follows:

```
VBoxManage setextradata "VM name" "VBoxInternal/TM/WarpDrivePercentage" 200
```

The above example will double the speed of the guest clock while

```
VBoxManage setextradata "VM name" "VBoxInternal/TM/WarpDrivePercentage" 50
```

will halve the speed of the guest clock. Note that changing the rate of the virtual clock can confuse the guest and can even lead to abnormal guest behavior. For instance, a higher clock rate means shorter timeouts for virtual devices with the result that a slightly increased response time of a virtual device due to an increased host load can cause guest failures. Note further that any time synchronization mechanism will frequently try to resynchronize the guest clock with the reference clock (which is the host clock if the VirtualBox Guest Additions are active). Therefore any time synchronization should be disabled if the rate of the guest clock is changed as described above (see chapter 9.14.3, *Tuning the Guest Additions time synchronization parameters*, page 206).

9.14.3 Tuning the Guest Additions time synchronization parameters

The VirtualBox Guest Additions ensure that the guest's system time is synchronized with the host time. There are several parameters which can be tuned. The parameters can be set for a specific VM using the following command:

```
VBoxManage guestproperty set "VM name" "/VirtualBox/GuestAdd/VBoxService/PARAMETER" VALUE
```

where `PARAMETER` is one of the following:

`--timesync-interval` Specifies the interval at which to synchronize the time with the host. The default is 10000 ms (10 seconds).

--timesync-min-adjust The minimum absolute drift value measured in milliseconds to make adjustments for. The default is 1000 ms on OS/2 and 100 ms elsewhere.

--timesync-latency-factor The factor to multiply the time query latency with to calculate the dynamic minimum adjust time. The default is 8 times, that means in detail: Measure the time it takes to determine the host time (the guest has to contact the VM host service which may take some time), multiply this value by 8 and do an adjustment only if the time difference between host and guest is bigger than this value. Don't do any time adjustment otherwise.

--timesync-max-latency The max host timer query latency to accept. The default is 250 ms.

--timesync-set-threshold The absolute drift threshold, given as milliseconds where to start setting the time instead of trying to smoothly adjust it. The default is 20 minutes.

--timesync-set-start Set the time when starting the time sync service.

--timesync-set-on-restore 0|1 Set the time after the VM was restored from a saved state when passing 1 as parameter (default). Disable by passing 0. In the latter case, the time will be adjusted smoothly which can take a long time.

All these parameters can be specified as command line parameters to VBoxService as well.

9.14.4 Disabling the Guest Additions time synchronization

Once installed and started, the VirtualBox Guest Additions will try to synchronize the guest time with the host time. This can be prevented by forbidding the guest service from reading the host clock:

```
VBoxManage setextradata "VM name" "VBoxInternal/Devices/VMMDev/0/Config/GetHostTimeDisabled" 1
```

9.15 Installing the alternate bridged networking driver on Solaris 11 hosts

Starting with VirtualBox 4.1, VirtualBox ships a new network filter driver that utilizes Solaris 11's Crossbow functionality. By default, this new driver is installed for Solaris 11 hosts (builds 159 and above) that has support for it.

To force installation of the older STREAMS based network filter driver, execute as root the following command before installing the VirtualBox package:

```
touch /etc/vboxinst_vboxflt
```

To force installation of the Crossbow based network filter driver, execute as root the following command before installing the VirtualBox package:

```
touch /etc/vboxinst_vboxbow
```

To check which driver is currently being used by VirtualBox, execute:

```
modinfo | grep vbox
```

If the output contains "vboxbow", it indicates VirtualBox is using the Crossbow network filter driver, while the name "vboxflt" indicates usage of the older STREAMS network filter.

9.16 VirtualBox VNIC templates for VLANs on Solaris 11 hosts

VirtualBox supports VNIC (Virtual Network Interface) templates for configuring VMs over VLANs.[3] A VirtualBox VNIC template is a VNIC whose name starts with "vboxvnic_template" (case-sensitive).

On Solaris 11 hosts[4], a VNIC template may be used to specify the VLAN ID to use while bridging over a network link.

Here is an example of how to use a VNIC template to configure a VM over a VLAN. Create a VirtualBox VNIC template, by executing as root:

```
dladm create-vnic -t -l nge0 -v 23 vboxvnic_template0
```

This will create a temporary VNIC template over interface "nge0" with the VLAN ID 23. To create VNIC templates that are persistent across host reboots, skip the `-t` parameter in the above command. You may check the current state of links using:

```
$ dladm show-link
LINK        CLASS     MTU    STATE    BRIDGE     OVER
nge0        phys      1500   up       --         --
nge1        phys      1500   down     --         --
vboxvnic_template0 vnic 1500 up       --         nge0

$ dladm show-vnic
LINK         OVER         SPEED  MACADDRESS        MACADDRTYPE         VID
vboxvnic_template0 nge0   1000   2:8:20:25:12:75   random              23
```

Once the VNIC template is created, any VMs that need to be on VLAN 23 over the interface "nge0" can be configured to bridge using this VNIC template.

VNIC templates makes managing VMs on VLANs simpler and efficient. The VLAN details are not stored as part of every VM's configuration but rather inherited from the VNIC template while starting the VM. The VNIC template itself can be modified anytime using `dladm`.

VNIC templates can be created with additional properties such as bandwidth limits, CPU fanout etc. Refer to your Solaris network documentation on how to accomplish this. These additional properties, if any, are also applied to VMs which bridge using the VNIC template.

9.17 Configuring multiple host-only network interfaces on Solaris hosts

By default VirtualBox provides you with one host-only network interface. Adding more host-only network interfaces on Solaris hosts requires manual configuration. Here's how to add another host-only network interface.

Begin by stopping all running VMs. Then, unplumb the existing "vboxnet0" interface by execute the following command as root:

```
ifconfig vboxnet0 unplumb
```

If you have several vboxnet interfaces, you will need to unplumb all of them. Once all vboxnet interfaces are unplumbed, remove the driver by executing the following command as root:

```
rem_drv vboxnet
```

Edit the file `/platform/i86pc/kernel/drv/vboxnet.conf` and add a line for the new interface we want to add as shown below:

[3] Support for Crossbow based bridged networking was introduced with VirtualBox 4.1 and requires Solaris 11 build 159 or above.

[4] When Crossbow based bridged networking is used.

```
name="vboxnet" parent="pseudo" instance=1;
name="vboxnet" parent="pseudo" instance=2;
```

Add as many of these lines as required with each line having a unique instance number.

Next, reload the vboxnet driver by executing the following command as root:

```
add_drv vboxnet
```

On Solaris 11.1 and newer hosts you may want to rename the default vanity interface name. To check what name has been assigned, execute:

```
dladm show-phys
LINK              MEDIA                STATE      SPEED  DUPLEX    DEVICE
net0              Ethernet             up         100    full      e1000g0
net2              Ethernet             up         1000   full      vboxnet1
net1              Ethernet             up         1000   full      vboxnet0
```

In the above example, we can rename "net2" to "vboxnet1" before proceeding to plumb the interface. This can be done by executing as root:

```
dladm rename-link net2 vboxnet1
```

Now plumb all the interfaces using `ifconfig vboxnetX plumb` (where 'X' would be 1 in this case). Once the interface is plumbed, it may be configured like any other network interface. Refer to the `ifconfig` documentation for further details.

To make the newly added interfaces' settings persistent across reboots, you will need to edit the files `/etc/inet/netmasks`, and if you are using NWAM `/etc/nwam/llp` and add the appropriate entries to set the netmask and static IP for each of those interfaces. The VirtualBox installer only updates these configuration files for the one "vboxnet0" interface it creates by default.

9.18 Configuring the VirtualBox CoreDumper on Solaris hosts

VirtualBox is capable of producing its own core files for extensive debugging when things go wrong. Currently this is only available on Solaris hosts.

The VirtualBox CoreDumper can be enabled using the following command:

```
VBoxManage setextradata "VM name" VBoxInternal2/CoreDumpEnabled 1
```

You can specify which directory to use for core dumps with this command:

```
VBoxManage setextradata "VM name" VBoxInternal2/CoreDumpDir <path-to-directory>
```

Make sure the directory you specify is on a volume with sufficient free space and that the VirtualBox process has sufficient permissions to write files to this directory. If you skip this command and don't specify any core dump directory, the current directory of the VirtualBox executable will be used (which would most likely fail when writing cores as they are protected with root permissions). It is recommended you explicitly set a core dump directory.

You must specify when the VirtualBox CoreDumper should be triggered. This is done using the following commands:

```
VBoxManage setextradata "VM name" VBoxInternal2/CoreDumpReplaceSystemDump 1
VBoxManage setextradata "VM name" VBoxInternal2/CoreDumpLive 1
```

At least one of the above two commands will have to be provided if you have enabled the VirtualBox CoreDumper.

Setting `CoreDumpReplaceSystemDump` sets up the VM to override the host's core dumping mechanism and in the event of any crash only the VirtualBox CoreDumper would produce the core file.

Setting `CoreDumpLive` sets up the VM to produce cores whenever the VM process receives a `SIGUSR2` signal. After producing the core file, the VM will not be terminated and will continue to run. You can thus take cores of the VM process using:

```
kill -s SIGUSR2 <VM-process-id>
```

Core files produced by the VirtualBox CoreDumper are of the form `core.vb.<ProcessName>.<ProcessID>`, for example `core.vb.VBoxHeadless.11321`.

9.19 VirtualBox and Solaris kernel zones

Solaris kernel zones on x86-based systems make use of hardware-assisted virtualization features like VirtualBox does. However, for kernel zones and VirtualBox to share this hardware resource, they need to co-operate.

By default, due to performance reasons, VirtualBox acquires the hardware-assisted virtualization resource (VT-x/AMD-V) globally on the host machine and uses it until the last VirtualBox VM that requires it is powered off. This prevents other software from using VT-x/AMD-V during the time VirtualBox has taken control of it.

VirtualBox can be instructed to relinquish use of hardware-assisted virtualization features when not executing guest code, thereby allowing kernel zones to make use of them. To do this, shutdown all VirtualBox VMs and execute the following command:

```
VBoxManage setproperty hwvirtexclusive off
```

This command needs to be executed only once as the setting is stored as part of the global VirtualBox settings which will continue to persist across host-reboots and VirtualBox upgrades.

9.20 Locking down the VirtualBox GUI

9.20.1 Customizing the VM manager

There are several advanced customization settings for locking down the VirtualBox manager, that is, removing some features that the user should not see.

```
VBoxManage setextradata global GUI/Customizations OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`noSelector` Don't allow to start the VirtualBox manager. Trying to do so will show a window containing a proper error message.

`noMenuBar` VM windows will not contain a menu bar.

`noStatusBar` VM windows will not contain a status bar.

To disable any of these VM manager customizations do

```
VBoxManage setextradata global GUI/Customizations
```

9.20.2 VM selector customization

The following per-machine VM extradata settings can be used to change the behavior of the VM selector window in respect of certain VMs:

```
VBoxManage setextradata "VM name" SETTING true
```

where `SETTING` can be:

`GUI/HideDetails` Don't show the VM configuration of a certain VM. The details window will remain just empty if this VM is selected.

`GUI/PreventReconfiguration` Don't allow the user to open the settings dialog for a certain VM.

`GUI/PreventSnapshotOperations` Prevent snapshot operations for a VM from the GUI, either at runtime or when the VM is powered off.

`GUI/HideFromManager` Hide a certain VM in the VM selector window.

`GUI/PreventApplicationUpdate` Disable the automatic update check and hide the corresponding menu item.

Please note that these settings wouldn't prevent the user from reconfiguring the VM by `VBoxManage modifyvm`.

9.20.3 Configure VM selector menu entries

You can disable (i.e. black-list) certain entries in the global settings page of the VM selector:

```
VBoxManage setextradata global GUI/RestrictedGlobalSettingsPages OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`General` Don't show the *General* settings pane.

`Input` Don't show the *Input* settings pane.

`Update` Don't show the *Update* settings pane.

`Language` Don't show the *Language* settings pane.

`Display` Don't show the *Display* settings pane.

`Network` Don't show the *Network* settings pane.

`Extensions` Don't show the *Extensions* settings pane.

`Proxy` Don't show the *Proxy* settings pane.

This is a global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata global GUI/RestrictedGlobalSettingsPages
```

9.20.4 Configure VM window menu entries

You can disable (i.e. black-list) certain menu actions in the VM window:

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeMenus OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All` Don't show any menu in the VM window.

`Machine` Don't show the *Machine* menu in the VM window.

`View` Don't show the *View* menu in the VM window.

`Devices` Don't show the *Devices* menu in the VM window.

`Help` Don't show the *Help* menu in the VM window.

`Debug` Don't show the *Debug* menu in the VM window. The debug menu is only visible if the GUI was started with special command line parameters or environment variable settings.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeMenus
```

You can also disable (i.e. blacklist) certain menu actions of certain menus. Use the following command to disable certain actions of the *Application* menu (only available on Mac OS X hosts):

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeApplicationMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All` Don't show any menu item in this menu.

`About` Don't show the *About* menu item in this menu.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeMenus
```

Use the following command to disable certain actions of the *Machine* menu:

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeApplicationMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All` Don't show any menu item in this menu.

`SettingsDialog` Don't show the *Settings* menu item in this menu.

`TakeSnapshot` Don't show the *Take Snapshot* menu item in this menu.

`TakeScreenshot` Don't show the *Take Screenshot* menu item in this menu.

`InformationDialog` Don't show the *Session Information* menu item in this menu.

`MouseIntegration` Don't show the *Disable Mouse Integration* menu item in this menu.

`TypeCAD` Don't show the *Insert Ctrl+Alt+Del* menu item in this menu.

TypeCABS Don't show the *Insert Ctrl+Alt+Backspace* menu item in this menu (available on X11 hosts only).

Pause Don't show the *Pause* menu item in this menu.

Reset Don't show the *Reset* menu item in this menu.

SaveState Don't show the *Save the machine state* menu item in this menu.

Shutdown Don't show the *ACPI Shutdown* menu item in this menu.

PowerOff Don't show the *Power Off the machine* menu item in this menu.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeApplicationMenuActions
```

Use the following command to disable certain actions of the *View* menu:

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeViewMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

All Don't show any menu item in this menu.

Fullscreen Don't show the *Switch to Fullscreen* menu item in this menu.

Seamless Don't show the *Switch to Seamless Mode* menu item in this menu.

Scale Don't show the *Switch to Scaled Mode* menu item in this menu.

GuestAutoresize Don't show the *Auto-resize Guest Display* menu item in this menu.

AdjustWindow Don't show the *Adjust Window Size* menu item in this menu.

Multiscreen Don't show the *Multiscreen* menu item in this menu (only visible in full screen / seamless mode).

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeViewMenuActions
```

Use the following command to disable certain actions of the *View* menu:

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeDevicesMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords to disable actions in the *Devices* menu:

All Don't show any menu item in this menu.

OpticalDevices Don't show the *CD/DVD Devices* menu item in this menu.

FloppyDevices Don't show the *FLoppy Devices* menu item in this menu.

USBDevices Don't show the *USB Devices* menu item in this menu.

SharedClipboard Don't show the *Shared Clipboard* menu item in this menu.

DragAndDrop Don't show the *Drag and Drop* menu item in this menu.

NetworkSettings Don't show the *Network Settings...* menu item in this menu.

SharedFoldersSettings Don't show the *Shared Folders Settings...* menu item in this menu.

VRDEServer Don't show the *Remove Display* menu item in this menu.

InstallGuestTools Don't show the *Insert Guest Additions CD imnage...* menu item in this menu.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeDevicesMenuActions
```

Use the following command to disable certain actions of the *View* menu:

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeDebuggerMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords to disable actions in the *Debug* menu (normally completely disabled):

All Don't show any menu item in this menu.

Statistics Don't show the *Statistics...* menu item in this menu.

CommandLine Don't show the *Command Line...* menu item in this menu.

Logging Don't show the *Logging...* menu item in this menu.

LogDialog Don't show the *Show Log...* menu item in this menu.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeDebuggerMenuActions
```

Use the following command to disable certain actions of the *View* menu:

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeHelpMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords to disable actions in the *Help* menu (normally completely disabled):

All Don't show any menu item in this menu.

Contents Don't show the *Contents...* menu item in this menu.

WebSite Don't show the *VirtualBox Web Site...* menu item in this menu.

ResetWarnings Don't show the *Reset All Warnings* menu item in this menu.

NetworkAccessManager Don't show the *Network Operations Manager* menu item in this menu.

About Don't show the *About* menu item in this menu (only on non Mac OS X hosts).

Contents Don't show the *Contents...* menu item in this menu.

Contents Don't show the *Contents...* menu item in this menu.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedRuntimeHelpMenuActions
```

9.20.5 Configure VM window status bar entries

You can disable (i.e. black-list) certain status bar items:

```
VBoxManage setextradata "VM name" GUI/RestrictedStatusBarIndicators OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

HardDisks Don't show the hard disk icon in the VM window status bar. By default the hard disk icon is only shown if the VM configuration contains one or more hard disks.

OpticalDisks Don't show the CD icon in the VM window status bar. By default the CD icon is only shown if the VM configuration contains one or more CD drives.

FloppyDisks Don't show the floppy icon in the VM window status bar. By default the floppy icon is only shown if the VM configuration contains one or more floppy drives.

Network Don't show the network icon in the VM window status bar. By default the network icon is only shown if the VM configuration contains one or more active network adapters.

USB Don't show the USB icon in the status bar.

SharedFolders Don't show the shared folders icon in the status bar.

VideoCapture Don't show the video capture icon in the status bar.

Features Don't show the CPU features icon in the status bar.

Mouse Don't show the mouse icon in the status bar.

Keyboard Don't show the keyboard icon in the status bar.

This is a per-VM setting. Any combination of the above is allowed. If all options are specified, no icons are displayed in the status bar of the VM window. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedStatusBarIndicators
```

9.20.6 Configure VM window visual modes

You can disable (i.e. black-list) certain VM visual modes:

```
VBoxManage setextradata "VM name" GUI/RestrictedVisualStates OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

Fullscreen Don't allow to switch the VM into full screen mode.

Seamless Don't allow to switch the VM into seamless mode.

Scale Don't allow to switch the VM into scale mode.

This is a per-VM setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name" GUI/RestrictedVisualStates
```

9.20.7 Host Key customization

To disable all host key combinations, open the preferences and change the host key to *None*. This might be useful when using VirtualBox in a kiosk mode.

To redefine or disable certain host key actions, use the following command:

```
VBoxManage setextradata global GUI/Input/MachineShortcuts "FullscreenMode=F,...."
```

The following list shows the possible host key actions together with their default host key shortcut. Setting an action to *None* will disable that host key action.

Action	Default Key	Action
`TakeSnapshot`	T	take a snapshot
`TakeScreenshot`	E	take a screenshot
`MouseIntegration`	I	toggle mouse integration
`TypeCAD`	Del	inject Ctrl+Alt+Del
`TypeCABS`	Backspace	inject Ctrl+Alt+Backspace
`Pause`	P	Pause the VM
`Reset`	R	(hard) reset the guest
`SaveState`		save the VM state and terminate the VM
`Shutdown`	H	press the (virtual) ACPI power button
`PowerOff`		power the VM off (without saving the state!)
`Close`	Q	show the VM close dialog
`FullscreenMode`	F	switch the VM into full screen
`SeamlessMode`	L	switch the VM into seamless mode
`ScaleMode`	C	switch the VM into scale mode
`GuestAutoResize`	G	automatically resize the guest window
`WindowAdjust`	A	immediately resize the guest window
`PopupMenu`	Home	show popup menu in full screen / seaml. mode
`SettingsDialog`	S	open the VM settings dialog
`InformationDialog`	N	show the VM information window
`NetworkAdaptersDialog`		show the VM network adapters dialog
`SharedFoldersDialog`		show the VM shared folders dialog
`InstallGuestAdditions`	D	mount the ISO containing the Guest Additions

To disable the full screen mode as well as the seamless mode, use the following command:

```
VBoxManage setextradata global GUI/Input/MachineShortcuts "FullscreenMode=None,SeamlessMode=None"
```

9.20.8 Action when terminating the VM

You can disallow (i.e. black-list) certain actions when terminating a VM. To disallow specific actions, type:

```
VBoxManage setextradata "VM name" GUI/RestrictedCloseActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`SaveState` Don't allow the user to save the VM state when terminating the VM.

`Shutdown` Don't allow the user to shutdown the VM by sending the ACPI power-off event to the guest.

`PowerOff` Don't allow the user to power off the VM.

`PowerOffRestoringSnapshot` Don't allow the user to return to the last snapshot when powering off the VM.

`Detach` Don't allow the user to detach from the VM process if the VM was started in separate mode.

This is a per-VM setting. Any combination of the above is allowed. If all options are specified, the VM cannot be shut down at all.

9.20.9 Default action when terminating the VM

You can define a specific action for terminating a VM. In contrast to the setting decribed in the previous section, this setting allows only one action when the user terminates the VM. No exit menu is shown.

```
VBoxManage setextradata "VM name" GUI/DefaultCloseAction ACTION
```

where `ACTION` is one of the following keywords:

`SaveState` Save the VM state before terminating the VM process.

`Shutdown` The VM is shut down by sending the ACPI power-off event to the guest.

`PowerOff` The VM is powered off.

`PowerOffRestoringSnapshot` The VM is powered off and the saved state returns to the last snapshot.

`Detach` Terminate the frontend but leave the VM process running.

This is a per-VM setting. Any combination of the above is allowed. If all options are specified, the VM cannot be shut down at all.

9.20.10 Action for handling a Guru Meditation

A VM runs into a Guru Meditation if there is a problem which cannot be fixed by other means than terminating the process. The default is to show a message window which instructs the user to open a bug report.

This behavior can be configured:

```
VBoxManage setextradata "VM name" GUI/GuruMeditationHandler MODE
```

where `MODE` is one of the following keywords:

`Default` A message window is shown. After the user confirmed, the VM is terminated.

`PowerOff` The VM is immediately powered-off without showing any message window. The VM logfile will show information about what happened.

`Ignore` The VM is left in stuck mode. Execution is stopped but no message window is shown. The VM has to be powered off manually.

This is a per-VM setting.

9.20.11 Configuring automatic mouse capturing

By default, the mouse is captured if the user clicks on the guest window and the guest expects relative mouse coordinates at this time. This happens if the pointing device is configured as PS/2 mouse and the guest did not (yet) start the VirtualBox Guest Additions (for instance, the guest is booting or no Guest Additions installed at all) or if the pointing device is configured as USB tablet but the guest has no USB driver loaded yet. Once the Guest Additions become active or the USB guest driver is started, the mouse capture is automatically released.

The default behavior is sometimes not desired. Therefore it can be configured:

```
VBoxManage setextradata "VM name" GUI/MouseCapturePolicy MODE
```

where `MODE` is one of the following keywords:

`Default` The default behavior as described above.

`HostComboOnly` The mouse is only captured if the Host Key is toggled.

`Disabled` The mouse is never captured, also not by toggling the Host Key

This is a per-VM setting.

9.20.12 Configuring automatic mouse capturing

By default, the mouse is captured if the user clicks on the guest window and the guest expects relative mouse coordinates at this time. This happens if the pointing device is configured as PS/2 mouse and the guest did not (yet) start the VirtualBox Guest Additions (for instance, the guest is booting or no Guest Additions installed at all) or if the pointing device is configured as USB tablet but the guest has no USB driver loaded yet. Once the Guest Additions become active or the USB guest driver is started, the mouse capture is automatically released.

The default behavior is sometimes not desired. Therefore it can be configured:

```
VBoxManage setextradata "VM name" GUI/MouseCapturePolicy MODE
```

where `MODE` is one of the following keywords:

`Default` The default behavior as described above.

`HostComboOnly` The mouse is only captured if the Host Key is toggled.

`Disabled` The mouse is never captured, also not by toggling the Host Key

This is a per-VM setting.

9.20.13 Requesting legacy full-screen mode

As of version 4.3.16, VirtualBox uses special window manager facilities to switch a multi-screen machine to full-screen on a multi-monitor host system. However, not all window managers provide these facilities correctly, so VirtualBox can be told to use the old method of switching to full-screen mode instead using the command:

```
VBoxManage setextradata global GUI/Fullscreen/LegacyMode true
```

You can go back to the new method using the command:

```
VBoxManage setextradata global GUI/Fullscreen/LegacyMode
```

This is a global setting.

9.21 Starting the VirtualBox web service automatically

The VirtualBox web service (`vboxwebsrv`) is used for controlling VirtualBox remotely. It is documented in detail in the VirtualBox Software Development Kit (SDK); please see chapter 11, *VirtualBox programming interfaces*, page 245. As the client base using this interface is growing, we added start scripts for the various operation systems we support. The following sections describe how to use them. The VirtualBox web service is never started automatically as a result of a standard installation.

9.21.1 Linux: starting the webservice via `init`

On Linux, the web service can be automatically started during host boot by adding appropriate parameters to the file `/etc/default/virtualbox`. There is one mandatory parameter, `VBOXWEB_USER`, which must be set to the user which will later start the VMs. The parameters in the table below all start with `VBOXWEB_` (`VBOXWEB_HOST`, `VBOXWEB_PORT` etc.):

Parameter	Description	Default
USER	The user as which the web service runs	
HOST	The host to bind the web service to	local-host
PORT	The port to bind the web service to	18083
SSL_KEYFILE	Server key and certificate file, PEM format	
SSL_PASSWORDFILE	File name for password to server key	
SSL_CACERT	CA certificate file, PEM format	
SSL_CAPATH	CA certificate path	
SSL_DHFILE	DH file name or DH key length in bits	
SSL_RANDFILE	File containing seed for random number generator	
TIMEOUT	Session timeout in seconds; 0 disables timeouts	300
CHECK_INTERVAL	Frequency of timeout checks in seconds	5
THREADS	Maximum number of worker threads to run in parallel	100
KEEPALIVE	Maximum number of requests before a socket will be closed	100
ROTATE	Number of log files; 0 disables log rotation	10
LOGSIZE	Maximum size of a log file in bytes to trigger rotation	1MB
LOGINTERVAL	Maximum time interval in seconds to trigger log rotation	1 day

Setting the parameter SSL_KEYFILE enables the SSL/TLS support. Using encryption is strongly encouraged, as otherwise everything (including passwords) is transferred in clear text.

9.21.2 Solaris: starting the web service via SMF

On Solaris hosts, the VirtualBox web service daemon is integrated into the SMF framework. You can change the parameters, but don't have to if the defaults below already match your needs:

```
svccfg -s svc:/application/virtualbox/webservice:default setprop config/host=localhost
svccfg -s svc:/application/virtualbox/webservice:default setprop config/port=18083
svccfg -s svc:/application/virtualbox/webservice:default setprop config/user=root
```

The table in the previous section showing the parameter names and defaults also applies to Solaris. The parameter names must be changed to lowercase and a prefix of `config/` has to be added, e.g. `config/user` or `config/ssl_keyfile`. If you made any change, don't forget to run the following command to put the changes into effect immediately:

```
svcadm refresh svc:/application/virtualbox/webservice:default
```

If you forget the above command then the previous settings will be used when enabling the service. Check the current property settings with:

```
svcprop -p config svc:/application/virtualbox/webservice:default
```

When everything is configured correctly you can start the VirtualBox web service with the following command:

```
svcadm enable svc:/application/virtualbox/webservice:default
```

For more information about SMF, please refer to the Solaris documentation.

9.21.3 Mac OS X: starting the webservice via launchd

On Mac OS X, launchd is used to start the VirtualBox webservice. An example configuration file can be found in `$HOME/Library/LaunchAgents/org.virtualbox.vboxwebsrv.plist`. It can be enabled by changing the `Disabled` key from `true` to `false`. To manually start the service use the following command:

```
launchctl load ~/Library/LaunchAgents/org.virtualbox.vboxwebsrv.plist
```

For additional information on how launchd services could be configured see https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/CreatingLaunchdJobs.html.

9.22 VirtualBox Watchdog

Starting with VirtualBox 4.2 the memory ballooning service formerly known as `VBoxBalloonCtrl` was renamed to VBoxWatchdog, which now incorporates several host services that are meant to be run in a server environment.

These services are:

- Memory ballooning control, which automatically takes care of a VM's configured memory balloon (see chapter 4.9.1, *Memory ballooning*, page 76 for an introduction to memory ballooning). This especially is useful for server environments where VMs may dynamically require more or less memory during runtime.

  The service periodically checks a VM's current memory balloon and its free guest RAM and automatically adjusts the current memory balloon by inflating or deflating it accordingly. This handling only applies to running VMs having recent Guest Additions installed.

- Host isolation detection, which provides a way to detect whether the host cannot reach the specific VirtualBox server instance anymore and take appropriate actions, such as shutting down, saving the current state or even powering down certain VMs.

All configuration values can be either specified via command line or global extradata, whereas command line values always have a higher priority when set. Some of the configuration values also be specified on a per-VM basis. So the overall lookup order is: command line, per-VM basis extradata (if available), global extradata.

9.22.1 Memory ballooning control

The memory ballooning control inflates and deflates the memory balloon of VMs based on the VMs free memory and the desired maximum balloon size.

To set up the memory ballooning control the maximum ballooning size a VM can reach needs to be set. This can be specified via command line with

```
--balloon-max <Size in MB>
```

, on a per-VM basis extradata value with

```
VBoxManage setextradata <VM-Name> VBoxInternal2/Watchdog/BalloonCtrl/BalloonSizeMax <Size in MB>
```

or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonSizeMax <Size in MB>
```

> **Note:** If no maximum ballooning size is specified by at least one of the parameters above, no ballooning will be performed at all.

Setting the ballooning increment in MB can be either done via command line with

```
--balloon-inc <Size in MB>
```

or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonIncrementMB <Size in MB>
```

Default ballooning increment is 256 MB if not specified.
Same goes with the ballooning decrement: Via command line with

```
--balloon-dec <Size in MB>
```

or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonDecrementMB <Size in MB>
```

Default ballooning decrement is 128 MB if not specified.
To define the lower limit in MB a balloon can be the command line with

```
--balloon-lower-limit <Size in MB>
```

can be used or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonLowerLimitMB <Size in MB>
```

is available. Default lower limit is 128 if not specified.

9.22.2 Host isolation detection

To detect whether a host is being isolated, that is, the host cannot reach the VirtualBox server instance anymore, the host needs to set an alternating value to a global extradata value within a time period. If this value is not set within that time period a timeout occurred and the so-called host isolation response will be performed to the VMs handled. Which VMs are handled can be controlled by defining VM groups and assigning VMs to those groups. By default no groups are set, meaning that all VMs on the server will be handled when no host response is received within 30 seconds.

To set the groups handled by the host isolation detection via command line:

```
--apimon-groups=<string[,stringN]>
```

or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/APIMonitor/Groups <string[,stringN]>
```

To set the host isolation timeout via command line:

```
--apimon-isln-timeout=<ms>
```

or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/APIMonitor/IsolationTimeoutMS <ms>
```

To set the actual host isolation response via command line:

```
--apimon-isln-response=<cmd>
```

or using a global extradata value with

```
VBoxManage setextradata global VBoxInternal2/Watchdog/APIMonitor/IsolationResponse <cmd>
```

The following response commands are available:

- `none`, which does nothing.
- `pause`, which pauses the execution of a VM.
- `poweroff`, which shuts down the VM by pressing the virtual power button. The VM will not have the chance of saving any data or veto the shutdown process.
- `save`, which saves the current machine state and powers off the VM afterwards. If saving the machine state fails the VM will be paused.
- `shutdown`, which shuts down the VM in a gentle way by sending an `ACPI` shutdown event to the VM's operating system. The OS then has the chance of doing a clean shutdown.

9.22.3 More information

For more advanced options and parameters like verbose logging check the built-in command line help accessible with `--help`.

9.22.4 Linux: starting the watchdog service via `init`

On Linux, the watchdog service can be automatically started during host boot by adding appropriate parameters to the file `/etc/default/virtualbox`. There is one mandatory parameter, `VBOXWATCHDOG_USER`, which must be set to the user which will later start the VMs. For backward compatibility you can also specify `VBOXBALLOONCTRL_USER`The parameters in the table below all start with `VBOXWATCHDOG_` (`VBOXWATCHDOG_BALLOON_INTERVAL`, `VBOXWATCHDOG_LOGSIZE` etc., and for previously existing parameters the `VBOXBALLOONCTRL_INTERVAL` etc. parameters can still be used):

Parameter	Description	Default
`USER`	The user as which the watchdog service runs	
`ROTATE`	Number of log files; 0 disables log rotation	10
`LOGSIZE`	Maximum size of a log file in bytes to trigger rotation	1MB
`LOGINTERVAL`	Maximum time interval in seconds to trigger log rotation	1 day
`BALLOON_INTERVAL`	Interval for checking the balloon size (msec)	30000
`BALLOON_INCREMENT`	Balloon size increment (MByte)	256
`BALLOON_DECREMENT`	Balloon size decrement (MByte)	128
`BALLOON_LOWERLIMIT`	Balloon size lower limit (MByte)	64
`BALLOON_SAFETYMARGIN`	Free memory required for decreasing the balloon size (MByte)	1024

9.22.5 Solaris: starting the watchdog service via SMF

On Solaris hosts, the VirtualBox watchdog service daemon is integrated into the SMF framework. You can change the parameters, but don't have to if the defaults already match your needs:

```
svccfg -s svc:/application/virtualbox/balloonctrl:default setprop config/balloon_interval=10000
svccfg -s svc:/application/virtualbox/balloonctrl:default setprop config/balloon_safetymargin=134217728
```

The table in the previous section showing the parameter names and defaults also applies to Solaris. The parameter names must be changed to lowercase and a prefix of `config/` has to be added, e.g. `config/user` or `config/balloon_safetymargin`. If you made any change, don't forget to run the following command to put the changes into effect immediately:

```
svcadm refresh svc:/application/virtualbox/balloonctrl:default
```

If you forget the above command then the previous settings will be used when enabling the service. Check the current property settings with:

```
svcprop -p config svc:/application/virtualbox/balloonctrl:default
```

When everything is configured correctly you can start the VirtualBox watchdog service with the following command:

```
svcadm enable svc:/application/virtualbox/balloonctrl:default
```

For more information about SMF, please refer to the Solaris documentation.

9.23 Other extension packs

Starting with VirtualBox 4.2.0 there is another extension pack, VNC, which is open source and replaces the previous integration of the VNC remote access protocol. This is experimental code, and will be initially available in the VirtualBox source code package only. It is to a large portion code contributed by users, and is not supported in any way by Oracle.

The keyboard handling is severely limited, and only the US keyboard layout works. Other keyboard layouts will have at least some keys which produce the wrong results (often quite surprising effects), and for layouts which have significant differences to the US keyboard layout it is most likely unusable.

It is possible to install both the Oracle VM VirtualBox Extension Pack and VNC, but only one VRDE module can be active at any time. The following command switches to the VNC VRDE module in VNC:

```
VBoxManage setproperty vrdeextpack VNC
```

Configuring the remote access works very similarly to VRDP (see chapter 7.1, *Remote display (VRDP support)*, page 104), with some limitations: VNC does not support specifying several port numbers, and the authentication is done differently. VNC can only deal with password authentication, and there is no option to use password hashes. This leaves no other choice than having a clear-text password in the VM configuration, which can be set with the following command:

```
VBoxManage modifyvm "VM name" --vrdeproperty VNCPassword=secret
```

The user is responsible for keeping this password secret, and it should be removed when a VM configuration is passed to another person, for whatever purpose. Some VNC servers claim to have "encrypted" passwords in the configuration. This is not true encryption, it is only concealing the passwords, which is exactly as secure as clear-text passwords.

The following command switches back to VRDP (if installed):

```
VBoxManage setproperty vrdeextpack "Oracle VM VirtualBox Extension Pack"
```

9.24 Starting virtual machines during system boot

Starting with VirtualBox 4.2.0 it is possible to start VMs automatically during system boot on Linux, Solaris and Mac OS X for all users.

9.24.1 Linux: starting the autostart service via `init`

On Linux, the autostart service is activated by setting two variables in `/etc/default/virtualbox`. The first one is `VBOXAUTOSTART_DB` which contains an absolute path to the autostart database directory. The directory should have write access for every user who should be able to start virtual machines automatically. Furthermore the directory should have the sticky bit set. The second variable is `VBOXAUTOSTART_CONFIG` which points the service to the autostart configuration file which is used during boot to determine whether to allow individual users to start a VM automatically and configure startup delays. The configuration file can be placed in `/etc/vbox` and contains several options. One is `default_policy` which controls whether the autostart service allows or denies to start a VM for users which are not in the exception list. The exception list starts with `exception_list` and contains a comma separated list with usernames. Furthermore a separate startup delay can be configured for every user to avoid overloading the host. A sample configuration is given below:

```
# Default policy is to deny starting a VM, the other option is "allow".
default_policy = deny

# Bob is allowed to start virtual machines but starting them
# will be delayed for 10 seconds
bob = {
    allow = true
    startup_delay = 10
}

# Alice is not allowed to start virtual machines, useful to exclude certain users
# if the default policy is set to allow.
alice = {
    allow = false
}
```

Every user who wants to enable autostart for individual machines has to set the path to the autostart database directory with

```
VBoxManage setproperty autostartdbpath <Autostart directory>
```

9.24.2 Solaris: starting the autostart service via SMF

On Solaris hosts, the VirtualBox autostart daemon is integrated into the SMF framework. To enable it you have to point the service to an existing configuration file which has the same format as on Linux (see chapter 9.24.1, *Linux: starting the autostart service via `init`*, page 223):

```
svccfg -s svc:/application/virtualbox/autostart:default setprop config/config=/etc/vbox/autostart.cfg
```

When everything is configured correctly you can start the VirtualBox autostart service with the following command:

```
svcadm enable svc:/application/virtualbox/autostart:default
```

For more information about SMF, please refer to the Solaris documentation.

9.24.3 Mac OS X: starting the autostart service via launchd

On Mac OS X, launchd is used to start the VirtualBox autostart service. An example configuration file can be found in `/Applications/VirtualBox.app/Contents/MacOS/org.virtualbox.vboxautostart.plist`. To enable the service copy the file to `/Library/LaunchDaemons` and change the `Disabled` key from `true` to `false`. Furthermore replace the second parameter to an existing configuration file which has the same format as on Linux (see chapter 9.24.1, *Linux: starting the autostart service via `init`*, page 223). To manually start the service use the following command:

```
launchctl load /Library/LaunchDaemons/org.virtualbox.vboxautostart.plist
```

For additional information on how launchd services could be configured see http://developer.apple.com/mac/library/documentation/MacOSX/Conceptual/BPSystemStartup/BPSystemStartup.html.

9.25 VirtualBox expert storage management

In case the snapshot model of VirtualBox is not sufficient it is possible to enable a special mode which makes it possible to reconfigure storage attachments while the VM is paused. The user has to make sure that the disk data stays consistent to the guest because unlike with hotplugging the guest is not informed about detached or newly attached media.

The expert storage management mode can be enabled per VM executing:

```
VBoxManage setextradata "VM name" "VBoxInternal2/SilentReconfigureWhilePaused" 1
```

Storage attachments can be reconfigured while the VM is paused afterwards using:

```
VBoxManage storageattach ...
```

9.26 Handling of host power management events

Some host power management events are handled by VirtualBox. The actual behavior depends on the platform:

Host Suspends This event is generated when the host is about to suspend, that is, the host saves the state to some non-volatile storage and powers off.

This event is currently only handled on Windows hosts and Mac OS X hosts. When this event is generated, VirtualBox will pause all running VMs.

Host Resumes This event is generated when the host woke up from the suspended state.

This event is currently only handled on Windows hosts and Mac OS X hosts. When this event is generated, VirtualBox will resume all VMs which are where paused before.

Battery Low The battery level reached a critical level (usually less than 5 percent charged).

This event is currently only handled on Windows hosts and Mac OS X hosts. When this event is generated, VirtualBox will save the state and terminate all VMs in preparation of a potential host powerdown.

The behavior can be configured. By executing the following command, no VM is saved:

```
VBoxManage setextradata global "VBoxInternal2/SavestateOnBatteryLow" 0
```

This is a global setting as well as a per-VM setting. The per-VM value has higher precedence than the global value. The following command will save the state of all VMs but will not save the state of VM “foo”:

```
VBoxManage setextradata global "VBoxInternal2/SavestateOnBatteryLow" 1
VBoxManage setextradata "foo" "VBoxInternal2/SavestateOnBatteryLow" 0
```

The first line is actually not required as by default the savestate action is performed.

9.27 Experimental support for passing through SSE4.1 / SSE4.2 instructions

To provide SSE 4.1 / SSE 4.2 support to guests, the host CPU has to implement these instruction sets. Starting with VirtualBox 4.3.8 it is possible to enable these instructions for certain guests using the following commands:

```
VBoxManage setextradata "VM name" VBoxInternal/CPUM/SSE4.1 1
VBoxManage setextradata "VM name" VBoxInternal/CPUM/SSE4.2 1
```

These are a per-VM settings and they are turned off by default.

9.28 Support for keyboard indicators synchronization

This feature makes the host keyboard lights match those of the virtual machine’s virtual keyboard when the machine window is selected. It is currently implemented for Mac OS X and Windows hosts and available as of releases 4.2.24 and 4.3.8. The feature can be enabled using the following command:

```
VBoxManage setextradata "VM name" GUI/HidLedsSync "1"
```

In order to disable it, use the same command but change “1” to “0”, or use the VBoxManage command to remove the extra data. This is a per-VM setting and it is disabled by default.

9.29 Capturing USB traffic for selected devices

Starting with VirtualBox 5.0 it is possible to capture USB traffic for single USB devices or on the root hub level which captures the traffic of all USB devices attached to the root hub. VirtualBox stores the traffic in a format which is compatible with Wireshark. To capture the traffic of a specific USB device it must be attached to the VM with VBoxManage using the following command:

```
VBoxManage controlvm "VM name" usbattach "device uuid|address" --capturefile "filename"
```

In order to enable capturing on the root hub use the following command while the VM is not running:

```
VBoxManage setextradata "VM name" VBoxInternal/Devices/usb-ehci/0/LUN#0/Config/CaptureFilename "filename"
```

The command above enables capturing on the root hub attached to the EHCI controller. To enable it for the OHCI or XHCI controller replace `usb-ehci` with `usb-ohci` or `usb-xhci` respectively.

9.30 Configuring the heartbeat service

VirtualBox ships a simple heartbeat service. Once the Guest Additions are active, the guest sends frequent heartbeat pings to the host. If the guest stops sending the heartbeat pings without properly terminating the service, the VM process will log this event in the VBox.log file. In the future it might be possible to configure dedicated actions but for now there is only a warning in the log file.

There are two parameters to configure. The *heartbeat interval* defines the time between two heartbeat pings. The default value is 2 seconds, that is, the heartbeat service of the VirtualBox Guest Additions will send a heartbeat ping every two seconds. The value in nanoseconds can be configured like this:

```
VBoxManage controlvm "VM name" VBoxInternal/Devices/VMMDev/0/Config/HeartbeatInterval 2000000000
```

The *heartbeat timeout* defines the time the host waits starting from the last heartbeat ping before it defines the guest as unresponsive. The default value is 2 times the heartbeat interval (4 seconds) and can be configured as following (in nanoseconds):

```
VBoxManage controlvm "VM name" VBoxInternal/Devices/VMMDev/0/Config/HeartbeatTimeout 4000000000
```

If the heartbeat timeout expires, there will be a log message like *VMMDev: HeartBeatCheckTimer: Guest seems to be unresponsive. Last heartbeat received 5 seconds ago.* If another heartbeat ping arrives after this warning, there will be a log message like *VMMDev: GuestHeartBeat: Guest is alive.*

9.31 Encryption of disk images

Starting with VirtualBox 5.0, it is possible to encrypt the data stored in hard disk images transparently for the guest. It does not depend on a specific image format to be used. Images which have the data encrypted are not portable between VirtualBox and other virtualization software.

VirtualBox uses the AES algorithm in XTS mode and supports 128 or 256 bit data encryption keys (DEK). The DEK is stored encrypted in the medium properties and is decrypted during VM startup by entering a password which was chosen when the image was encrypted.

Since the DEK is stored as part of the VM configuration file, it is important that it is kept safe. Losing the DEK means that the data stored in the disk images is lost irrecoverably. Having complete and up to date backups of all data related to the VM is the responsibility of the user.

9.31.1 Limitations

There are some limitations the user needs to be aware of when using this feature:

- This feature is part of the Oracle VM VirtualBox Extension Pack, which needs to be installed. Otherwise disk encryption is unavailable.
- Since encryption works only on the stored user data, it is currently not possible to check for metadata integrity of the disk image. Attackers might destroy data by removing or changing blocks of data in the image or change metadata items such as the disk size.
- Exporting appliances which contain encrypted disk images is not possible because the OVF specification doesn't support this. All images are therefore decrypted during export.
- The DEK is kept in memory while the VM is running to be able to decrypt data read and encrypt data written by the guest. While this should be obvious the user needs to be aware of this because an attacker might be able to extract the key on a compromised host and decrypt the data.
- When encrypting or decrypting the images, the password is passed in clear text via the VirtualBox API. This needs to be kept in mind, especially when using third party API clients which make use of the webservice where the password might be transmitted over the network. The use of HTTPS is mandatory in such a case.
- Encrypting images with differencing images is only possible if there are no snapshots or a linear chain of snapshots. This limitation may be addressed in a future VirtualBox version.

9.31.2 Encrypting disk images

Encrypting disk images can be done either using the GUI or VBoxManage. While the GUI is easier to use, it works on a per VM basis and encrypts all disk images attached to the specific VM. With VBoxManage one can encrypt individual images (including all differencing images). To encrypt an unencrypted medium with VBoxManage, use:

```
VBoxManage encryptmedium "uuid|filename" --newpassword "file|-" --cipher "cipher id" --newpasswordid "id"
```

To supply the encryption password point VBoxManage to the file where the password is stored or specify `-` to let VBoxManage ask you for the password on the command line.

The cipher parameter specifies the cipher to use for encryption and can be either `AES-XTS128-PLAIN64` or `AES-XTS256-PLAIN64`. The specified password identifier can be freely chosen by the user and is used for correct identification when supplying multiple passwords during VM startup.

If the user uses the same password when encrypting multiple images and also the same password identifier, the user needs to supply the password only once during VM startup.

9.31.3 Starting a VM with encrypted images

When a VM is started using the GUI, a dialog will open where the user needs to enter all passwords for all encrypted images attached to the VM. If another frontend like VBoxHeadless is used, the VM will be paused as soon as the guest tries to access an encrypted disk. The user needs to provide the passwords through VBoxManage using the following command:

```
VBoxManage controlvm "uuid|vmname" addencpassword "id" "password" [--removeonsuspend "yes|no"]
```

The `id` parameter must be the same as the password identifier supplied when encrypting the images. `password` is the password used when encrypting the images. The user can optionally specify `--removeonsuspend "yes|no"` to specify whether to remove the password from VM memory when the VM is suspended. Before the VM can be resumed, the user needs to supply the passwords again. This is useful when a VM is suspended by a host suspend event and the user doesn't want the password to remain in memory.

9.31.4 Decrypting encrypted images

In some circumstances it might be required to decrypt previously encrypted images. This can be done in the GUI for a complete VM or using VBoxManage with the following command:

```
VBoxManage encryptmedium "uuid|filename" --oldpassword "file|-"
```

The only required parameter is the password the image was encrypted with. The options are the same as for encrypting images.

9.32 Paravirtualized debugging

In this section we cover debugging of guest operating systems using interfaces supported by paravirtualization providers.

> **Note:** Paravirtualized debugging significantly alter guest operating system behaviour and should only be used by expert users for debugging and diagnostics.

These debug options are specified as a string of key-value pairs separated by commas. An empty string disables paravirtualized debugging.

9.32.1 Hyper-V debug options

All of the options listed below are optional, and thus the default value specified will be used when the corresponding key-value pair is not specified.

- Key: **`enabled`**

  Value: `0` or `1`

  Default: `0`

  Specify `1` to enable the Hyper-V debug interface. If this key-value pair is not specified or the value is not `1`, the Hyper-V debug interface is disabled regardless of other key-value pairs being present.

- Key: **`address`**

  Value: IPv4 address

  Default: 127.0.0.1

  Specify the IPv4 address where the remote debugger is connected.

- Key: **`port`**

  Value: UDP port number

  Default: 50000

  Specify the UDP port number where the remote debugger is connected.

- Key: **`vendor`**

  Value: Hyper-V vendor signature reported via CPUID to the guest

  Default: When debugging is enabled: `Microsoft Hv`, otherwise: `VBoxVBoxVBox`

  Specify the Hyper-V vendor signature which is exposed to the guest via CPUID. For debugging Microsoft Windows guests, it is required the hypervisor reports the Microsoft vendor.

- Key: **`hypercallinterface`**

  Value: `0` or `1`

  Default: `0`

  Specify whether hypercalls should be suggested for initiating debug data transfers between host and guest rather than MSRs when requested by the guest.

- Key: **`vsinterface`**

  Value: `0` or `1`

  Default: When debugging is enabled, `1`, otherwise `0`

  Specify whether to expose the "VS#1" (virtualization service) interface to the guest. This interface is required for debugging Microsoft Windows 10 32-bit guests, but is optional for other Windows versions.

9.32.1.1 Setting up Windows guests for debugging with the Hyper-V paravirtualization provider

Windows supports debugging over a serial cable, USB, IEEE 1394 Firewire, and Ethernet (only Windows 8 and later). USB and IEEE 1394 are not applicable for virtual machines, and Ethernet requires Windows 8 or later. While serial connection is universally usable, it is slow.

Debugging using the Hyper-V debug transport, supported on Windows Vista and later, offers significant benefits. It provides excellent performance due to direct host-to-guest transfers, it is easy to set up and requires minimal support from the hypervisor. It can be used with the debugger running on the same host as the VM or with the debugger and VM on separate machines connected over a network.

Prerequisites

- A VM configured for Hyper-V paravirtualization running a Windows Vista or newer Windows guest. You may check the effective paravirtualization provider for your VM from the output of the following VBoxManage command:

  ```
  VBoxManage showvminfo "VM name"
  ```

- A sufficiently up-to-date version of the Microsoft WinDbg debugger required to debug the version of Windows in your VM.

- While Windows 8 and newer Windows guests ship with Hyper-V debug support, Windows 7 and Vista do not. To use Hyper-V debugging with a Windows 7 or Vista guest, copy the file `kdvm.dll` from a Windows 8.0 installation[5]. This file is typically located in `C:\Windows\System32`. Copy it to the same location in your Windows 7/Vista guest. Make sure you copy the 32-bit or 64-bit version of the DLL which matches your guest OS.

[5] Only Windows 8.0 ships `kdvm.dll`. Windows 8.1 and newer Windows versions do not.

VM and guest configuration

1. Power off the VM.
2. Enable the debug options by executing the following VBoxManage command:

   ```
   VBoxManage modifyvm "VM name" --paravirtdebug "enabled=1"
   ```

   The above command assumes your debugger will connect to your host machine on UDP port 50000. However, if you need to run the debugger on a remote machine you may specify the remote address and port here, e.g. using:

   ```
   VBoxManage modifyvm "VM name" --paravirtdebug "enabled=1,address=192.168.32.1,port=55000"
   ```

   Refer chapter 9.32.1, *Hyper-V debug options*, page 228 for the complete set of options.
3. Start the VM.
4. In the guest, start an elevated command prompt and execute the following commands:
   - For a Windows 8 or newer Windows guest:

     ```
     bcdedit /dbgsettings net hostip:5.5.5.5 port:50000 key:1.2.3.4
     ```
   - For a Windows 7 or Vista guest:

     ```
     bcdedit /set loadoptions host_ip=5.5.5.5,host_port=50000,encryption_key=1.2.3.4
     bcdedit /set dbgtransport kdvm.dll
     ```

     The IP address and port in the `bcdedit` command are ignored when using the Hyper-V debug transport. Any valid IP and a port number greater than 49151 and lower than 65536 can be entered.

     The encryption key in the `bcdedit` command is relevant and must be valid. The key "1.2.3.4" used in the above example is valid and may be used if security is not a concern. If you do not specify any encryption key, `bcdedit` will generate one for you and you will need to copy this key to later enter in Microsoft WinDbg on the remote end. This encryption key is used to encrypt the debug data exchanged between Windows and the debugger.
   - Execute one or more of the following commands to enable debugging for the appropriate phase or component of your Windows guest:

     ```
     bcdedit /set debug on
     bcdedit /set bootdebug on
     bcdedit /set {bootmgr} bootdebug on
     ```

     Please note that the `bootdebug` options are only effective on Windows 8 or newer when using the Hyper-V debug transport. Refer to Microsoft Windows documentation for detailed explanation of `bcdedit` options.
5. Start Microsoft WinDbg on your host machine or remote host.

   From the "File" menu, select "Kernel debug". Under the "NET" tab, specify the UDP port number you used in the `paravirtdebug` options. If you didn't specify any, leave it as 50000. Ensure that the UDP port is not blocked by a firewall or other security software.

   In the "Key" field, enter `1.2.3.4` or the encryption key from the `bcdedit` command in your Windows guest.

   Now press "OK" to start listening for connections. Microsoft WinDbg typically shows a "Waiting to reconnect" message during this phase.

   Alternatively, launch WinDbg from the command line to directly start a debug session:

   ```
   windbg.exe -k net:port=50000,key=1.2.3.4
   ```

   Please refer to the WinDbg documentation for complete command line syntax.
6. Reboot your Windows guest and it should then connect as a debuggee with Microsoft WinDbg.

9.33 PC speaker passthrough

As an experimental feature (primarily due to being limited to Linux host only and unknown Linux distribution coverage) VirtualBox supports passing through the PC speaker to the host. The PC speaker (sometimes called system speaker) is a way to produce audible feedback such as beeps without the need for regular audio/sound card support.

The PC speaker passthrough feature in VirtualBox handles beeps only. Advanced PC speaker use by the VM (such as PCM audio) will not work, resulting in undefined host behavior.

Producing beeps on Linux is unfortunately a very complex topic. VirtualBox offers a collection of options, in an attempt to make this work deterministically and reliably on as many Linux distributions and system configurations as possible:

Code	Device	Notes
1	`/dev/input/by-path/platform-pcspkr-event-spkr`	Direct host PC speaker use.
2	`/dev/tty`	Uses the terminal association of the VM process. VM needs to be started on a virtual console.
3	`/dev/tty0` or `/dev/vc/0`	Can only be used by user `root` or users with capability `cap_sys_tty_config`
9	user specified console or evdev device path	Like 1-3, just with a custom device path.
70	`/dev/tty`	Standard beep only. Loses frequency and length. See code 2.
79	user specified terminal device path	Like 70, just with a custom device path.
100	all of the above	Tries all above codes.

To enable PC speaker passthrough use the following command:

```
VBoxManage setextradata "VM name" "VBoxInternal/Devices/i8254/0/Config/PassthroughSpeaker" N
```

Replace `N` with the code representing the case you want to use. Changing this setting will take effect when the VM is started next. It is safe to enable PC speaker passthrough on all host OSes. It will only have an effect on Linux.

The VM log file, `VBox.log`, will contain lines with the prefix `PIT: speaker:` showing the PC speaker passthrough setup activities. It gives hints which device it picked or why it failed.

Enabling PC speaker passthrough for the VM is usually the simple part. The real difficulty is making sure that VirtualBox can access the necessary device, because in a typical Linux install most of them can only be accessed by user `root`. You should follow the preferred way to persistently change this, e.g. by referring to your distribution's documentation. Since there are countless Linux distribution variants, we can only give the general hints that there is often a way to give the X11 session user access to additional devices, or you need to find a working solution using a udev configuration file. If everything fails you might try setting the permissions using a script which is run late enough in the host system startup.

Sometimes additional rules are applied by the kernel to limit access (e.g. that the VM process must have the same controlling terminal as the device configured to be used for beeping, something which is often very difficult to achieve for GUI applications such as VirtualBox). The table above contains some hints, but generally refer to the Linux documentation.

If you have trouble getting any beeps even if the device permissions are set up and VBox.log confirms that it uses evdev or console for the PC speaker control, check if your system has a PC speaker. Some systems do not have one. Other complications can arise from Linux rerouting the PC speaker output to a sound card. Check if the beeps are audible if you connect speakers to your sound card. Today almost all systems have one. Finally, check if the audio mixer control has a channel named "beep" (could be hidden in the mixer settings) and that it isn't muted.

9.34 Accessing USB devices exposed over the network with USB/IP

Starting with 5.1.0, VirtualBox supports passing through USB devices which are exposed over the network using the USB over IP protocol without the need to configure the client side provided by the kernel and usbip tools. Furthermore, this feature works with VirtualBox running on any supported host, rather than just Linux alone - as is the case with the official client.

To enable support for passing through USB/IP devices, the device server exporting the devices must be added with the following command:

```
VBoxManage usbdevsource add "Unique name" --backend "USBIP" --address "Device server[:port]"
```

USB devices exported on the device server are then accessible through the GUI or VBoxManage, like any USB devices attached locally. This can be used multiple times to access different device servers.

To remove a device server, the following command can be used:

```
VBoxManage usbdevsource remove "Unique name"
```

9.34.1 Setting up USB/IP support on a Linux system

This section gives a brief overview on how to set up a Linux based system to act as a USB device server. The system on the server requires that the `usbip-core.ko` and `usbip-host.ko` kernel drivers are available, and that the USB/IP tools package is installed. The particular installation method for the necessary tools depends on which distribution is used. For example, for Debian based systems - the following command should be used to install the required tools:

```
apt-get install usbip-utils
```

To check whether the necessary tools are already installed use the following command:

```
$ usbip list -l
```

which should produce output similar to that shown in the example below:

```
 - busid 4-2 (0bda:0301)
   Realtek Semiconductor Corp. : multicard reader (0bda:0301)

 - busid 5-1 (046d:c52b)
   Logitech, Inc. : Unifying Receiver (046d:c52b)
```

If everything is installed, the USB/IP server needs to be started as `root` using the following command:

```
usbipd -D
```

Refer to the documentation for the installed distribution to determine how to start the service when the system boots.

By default, no device on the server is exported - and this must be done manually for each device. To export a device use:

```
usbip bind -b "bus identifier"
```

To export the multicard reader from above, for example - use:

```
usbip bind -b 4-2
```

9.34.2 Security considerations

The communication between the server and client is unencrypted and there is no authorization required to access exported devices. An attacker might sniff sensitive data or gain control over a device. To mitigate this risk, the device should be exposed over a local network to which only trusted clients have access. To access the device remotely over a public network, a VPN solution should be used to provide the required level of security protection.

10 Technical background

The contents of this chapter are not required to use VirtualBox successfully. The following is provided as additional information for readers who are more familiar with computer architecture and technology and wish to find out more about how VirtualBox works "under the hood".

10.1 Where VirtualBox stores its files

In VirtualBox, a virtual machine and its settings are described in a virtual machine settings file in XML format. In addition, most virtual machine have one or more virtual hard disks, which are typically represented by disk images (e.g. in VDI format). Where all these files are stored depends on which version of VirtualBox created the machine.

10.1.1 Machines created by VirtualBox version 4.0 or later

Starting with version 4.0, by default, each virtual machine has one directory on your host computer where all the files of that machine are stored – the XML settings file (with a `.vbox` file extension) and its disk images.

By default, this "machine folder" is placed in a common folder called "VirtualBox VMs", which VirtualBox creates in the current system user's home directory. The location of this home directory depends on the conventions of the host operating system:

- On Windows, this is the location returned by the `SHGetFolderPath` function of the Windows system library Shell32.dll, asking for the user profile. Only on very old Windows versions which don't have this function or where it unexpectedly returns an error, there is a fallback based on environment variables: first `%USERPROFILE%` is checked, if it doesn't exist then an attempt with `%HOMEDRIVE%%HOMEPATH%` is made. Typical value is `C:\Users\username`.

- On Linux, Mac OS X and Solaris, this is generally taken from the environment variable `$HOME`, except for the user `root` for which it's taken from the account database (as a workaround for the frequent trouble caused by users using VirtualBox in combination with the tool `sudo` which by default doesn't reset the environment variable `$HOME`). Typical value on Linux and Solaris is `/home/username` and on Mac OS X `/Users/username`.

For simplicity, we will abbreviate this as `$HOME` below. Using that convention, the common folder for all virtual machines is `$HOME/VirtualBox VMs`.

As an example, when you create a virtual machine called "Example VM", you will find that VirtualBox creates

1. the folder `$HOME/VirtualBox VMs/Example VM/` and, in that folder,

2. the settings file `Example VM.vbox` and

3. the virtual disk image `Example VM.vdi`.

This is the default layout if you use the "Create new virtual machine" wizard as described in chapter 1.7, *Creating your first virtual machine*, page 18. Once you start working with the VM, additional files will show up: you will find log files in a subfolder called `Logs`, and once you have

taken snapshots, they will appear in a `Snapshots` subfolder. For each VM, you can change the location of its snapshots folder in the VM settings.

You can change the default machine folder by selecting "Preferences" from the "File" menu in the VirtualBox main window. Then, in the window that pops up, click on the "General" tab. Alternatively, use `VBoxManage setproperty machinefolder`; see chapter 8.30, *VBoxManage setproperty*, page 161.

10.1.2 Machines created by VirtualBox versions before 4.0

If you have upgraded to VirtualBox 4.0 from an earlier version of VirtualBox, you probably have settings files and disks in the earlier file system layout.

Before version 4.0, VirtualBox separated the machine settings files from virtual disk images. The machine settings files had an `.xml` file extension and resided in a folder called "Machines" under the global VirtualBox configuration directory (see the next section). So, for example, on Linux, this was the hidden `$HOME/.VirtualBox/Machines` directory. The default hard disks folder was called "HardDisks" and resided in the `.VirtualBox` folder as well. Both locations could be changed by the user in the global preferences. (The concept of a "default hard disk folder" has been abandoned with VirtualBox 4.0, since disk images now reside in each machine's folder by default.)

The old layout had several severe disadvantages.

1. It was very difficult to move a virtual machine from one host to another because the files involved did not reside in the same folder. In addition, the virtual media of all machines were registered with a global registry in the central VirtualBox settings file (`$HOME/.VirtualBox/VirtualBox.xml`).

   To move a machine to another host, it was therefore not enough to move the XML settings file and the disk images (which were in different locations), but the hard disk entries from the global media registry XML had to be meticulously copied as well, which was close to impossible if the machine had snapshots and therefore differencing images.

2. Storing virtual disk images, which can grow very large, under the hidden `.VirtualBox` directory (at least on Linux and Solaris hosts) made many users wonder where their disk space had gone.

Whereas new VMs created with VirtualBox 4.0 or later will conform to the new layout, for maximum compatibility, old VMs are *not* converted to the new layout. Otherwise machine settings would be irrevocably broken if a user downgraded from 4.0 back to an older version of VirtualBox.

10.1.3 Global configuration data

In addition to the files of the virtual machines, VirtualBox maintains global configuration data. On Linux and Solaris as of VirtualBox 4.3, this is in the hidden directory `$HOME/.config/VirtualBox`, although `$HOME/.VirtualBox` will be used if it exists for compatibility with earlier versions; on Windows (and on Linux and Solaris with VirtualBox 4.2 and earlier) this is in `$HOME/.VirtualBox`; on a Mac it resides in `$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, or additionally on Linux or Solaris by using the standard `XDG_CONFIG_HOME` variable. (Since the global `VirtualBox.xml` settings file points to all other configuration files, this allows for switching between several VirtualBox configurations entirely.)

Most importantly, in this directory, VirtualBox stores its global settings file, another XML file called `VirtualBox.xml`. This includes global configuration options and the list of registered

virtual machines with pointers to their XML settings files. (Neither the location of this file nor its directory has changed with VirtualBox 4.0.)

Before VirtualBox 4.0, all virtual media (disk image files) were also contained in a global registry in this settings file. For compatibility, this media registry still exists if you upgrade VirtualBox and there are media from machines which were created with a version before 4.0. If you have no such machines, then there will be no global media registry; with VirtualBox 4.0, each machine XML file has its own media registry.

Also before VirtualBox 4.0, the default "Machines" folder and the default "HardDisks" folder resided under the VirtualBox configuration directory (e.g. `$HOME/.VirtualBox/Machines` on Linux). If you are upgrading from a VirtualBox version before 4.0, files in these directories are not automatically moved in order not to break backwards compatibility.

10.1.4 Summary of 4.0 configuration changes

The following table gives a brief overview of the configuration changes between older versions and version 4.0 or above:

Setting	Before 4.0	4.0 or above
Default machines folder	`$HOME/.VirtualBox/Machines`	`$HOME/VirtualBox VMs`
Default disk image location	`$HOME/.VirtualBox/HardDisks`	In each machine's folder
Machine settings file extension	`.xml`	`.vbox`
Media registry	Global `VirtualBox.xml` file	Each machine settings file
Media registration	Explicit open/close required	Automatic on attach

10.1.5 VirtualBox XML files

VirtualBox uses XML for both the machine settings files and the global configuration file, `VirtualBox.xml`.

All VirtualBox XML files are versioned. When a new settings file is created (e.g. because a new virtual machine is created), VirtualBox automatically uses the settings format of the current VirtualBox version. These files may not be readable if you downgrade to an earlier version of VirtualBox. However, when VirtualBox encounters a settings file from an earlier version (e.g. after upgrading VirtualBox), it attempts to preserve the settings format as much as possible. It will only silently upgrade the settings format if the current settings cannot be expressed in the old format, for example because you enabled a feature that was not present in an earlier version of VirtualBox.[1] In such cases, VirtualBox backs up the old settings file in the virtual machine's configuration directory. If you need to go back to the earlier version of VirtualBox, then you will need to manually copy these backup files back.

We intentionally do not document the specifications of the VirtualBox XML 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*, page 114) and its API (see chapter 11, *VirtualBox programming interfaces*, page 245).

[1] As an example, before VirtualBox 3.1, it was only possible to enable or disable a single DVD drive in a virtual machine. If it was enabled, then it would always be visible as the secondary master of the IDE controller. With VirtualBox 3.1, DVD drives can be attached to arbitrary slots of arbitrary controllers, so they could be the secondary slave of an IDE controller or in a SATA slot. If you have a machine settings file from an earlier version and upgrade VirtualBox to 3.1 and then move the DVD drive from its default position, this cannot be expressed in the old settings format; the XML machine file would get written in the new format, and a backup file of the old format would be kept.

10.2 VirtualBox executables and components

VirtualBox was designed to be modular and flexible. When the VirtualBox graphical user interface (GUI) is opened and a VM is started, at least three processes are running:

1. `VBoxSVC`, the VirtualBox service process which always runs in the background. This process is started automatically by the first VirtualBox client process (the GUI, `VBoxManage`, `VBoxHeadless`, the web service or others) and exits a short time after the last client exits. The service is responsible for bookkeeping, maintaining the state of all VMs, and for providing communication between VirtualBox components. This communication is implemented via COM/XPCOM.

> **Note:** When we refer to "clients" here, we mean the local clients of a particular `VBoxSVC` server process, not clients in a network. VirtualBox employs its own client/server design to allow its processes to cooperate, but all these processes run under the same user account on the host operating system, and this is totally transparent to the user.

2. The GUI process, `VirtualBox`, a client application based on the cross-platform Qt library. When started without the `--startvm` option, this application acts as the VirtualBox manager, displaying the VMs and their settings. It then communicates settings and state changes to `VBoxSVC` and also reflects changes effected through other means, e.g., `VBoxManage`.

3. If the `VirtualBox` client application is started with the `--startvm` argument, it loads the VMM library which includes the actual hypervisor and then runs a virtual machine and provides the input and output for the guest.

Any VirtualBox front-end (client) will communicate with the service process and can both control and reflect the current state. For example, either the VM selector or the VM window or VBoxManage can be used to pause the running VM, and other components will always reflect the changed state.

The VirtualBox GUI application is only one of several available front ends (clients). The complete list shipped with VirtualBox is:

1. `VirtualBox`, the Qt front end implementing the manager and running VMs;

2. `VBoxManage`, a less user-friendly but more powerful alternative, described in chapter 8, *VBoxManage*, page 114.

3. `VBoxSDL`, a simple graphical front end based on the SDL library; see chapter 9.1, *VBoxSDL, the simplified VM displayer*, page 185.

4. `VBoxHeadless`, a VM front end which does not directly provide any video output and keyboard/mouse input, but allows redirection via VirtualBox Remote Desktop Extension; see chapter 7.1.2, *VBoxHeadless, the remote desktop server*, page 105.

5. `vboxwebsrv`, the VirtualBox web service process which allows for controlling a VirtualBox host remotely. This is described in detail in the VirtualBox Software Development Kit (SDK) reference; please see chapter 11, *VirtualBox programming interfaces*, page 245 for details.

6. The VirtualBox Python shell, a Python alternative to VBoxManage. This is also described in the SDK reference.

Internally, VirtualBox consists of many more or less separate components. You may encounter these when analyzing VirtualBox internal error messages or log files. These include:

- IPRT, a portable runtime library which abstracts file access, threading, string manipulation, etc. Whenever VirtualBox accesses host operating features, it does so through this library for cross-platform portability.
- VMM (Virtual Machine Monitor), the heart of the hypervisor.
- EM (Execution Manager), controls execution of guest code.
- REM (Recompiled Execution Monitor), provides software emulation of CPU instructions.
- TRPM (Trap Manager), intercepts and processes guest traps and exceptions.
- HM (Hardware Acceleration Manager), provides support for VT-x and AMD-V.
- GIM (Guest Interface Manager), provides support for various paravirtualization interfaces to the guest.
- PDM (Pluggable Device Manager), an abstract interface between the VMM and emulated devices which separates device implementations from VMM internals and makes it easy to add new emulated devices. Through PDM, third-party developers can add new virtual devices to VirtualBox without having to change VirtualBox itself.
- PGM (Page Manager), a component controlling guest paging.
- PATM (Patch Manager), patches guest code to improve and speed up software virtualization.
- TM (Time Manager), handles timers and all aspects of time inside guests.
- CFGM (Configuration Manager), provides a tree structure which holds configuration settings for the VM and all emulated devices.
- SSM (Saved State Manager), saves and loads VM state.
- VUSB (Virtual USB), a USB layer which separates emulated USB controllers from the controllers on the host and from USB devices; this also enables remote USB.
- DBGF (Debug Facility), a built-in VM debugger.
- VirtualBox emulates a number of devices to provide the hardware environment that various guests need. Most of these are standard devices found in many PC compatible machines and widely supported by guest operating systems. For network and storage devices in particular, there are several options for the emulated devices to access the underlying hardware. These devices are managed by PDM.
- Guest Additions for various guest operating systems. This is code that is installed from within a virtual machine; see chapter 4, *Guest Additions*, page 60.
- The "Main" component is special: it ties all the above bits together and is the only public API that VirtualBox provides. All the client processes listed above use only this API and never access the hypervisor components directly. As a result, third-party applications that use the VirtualBox Main API can rely on the fact that it is always well-tested and that all capabilities of VirtualBox are fully exposed. It is this API that is described in the VirtualBox SDK mentioned above (again, see chapter 11, *VirtualBox programming interfaces*, page 245).

10.3 Hardware vs. software virtualization

VirtualBox allows software in the virtual machine to run directly on the processor of the host, but an array of complex techniques is employed to intercept operations that would interfere with your host. Whenever the guest attempts to do something that could be harmful to your computer and its data, VirtualBox steps in and takes action. In particular, for lots of hardware that the guest believes to be accessing, VirtualBox simulates a certain "virtual" environment according to how you have configured a virtual machine. For example, when the guest attempts to access a hard disk, VirtualBox redirects these requests to whatever you have configured to be the virtual machine's virtual hard disk – normally, an image file on your host.

Unfortunately, the x86 platform was never designed to be virtualized. Detecting situations in which VirtualBox needs to take control over the guest code that is executing, as described above, is difficult. There are two ways in which to achieve this:

- Since 2006, Intel and AMD processors have had support for so-called **"hardware virtualization"**. This means that these processors can help VirtualBox to intercept potentially dangerous operations that a guest operating system may be attempting and also makes it easier to present virtual hardware to a virtual machine.

  These hardware features differ between Intel and AMD processors. Intel named its technology **VT-x**; AMD calls theirs **AMD-V**. The Intel and AMD support for virtualization is very different in detail, but not very different in principle.

  Note: On many systems, the hardware virtualization features first need to be enabled in the BIOS before VirtualBox can use them.

- As opposed to other virtualization software, for many usage scenarios, VirtualBox does not *require* hardware virtualization features to be present. Through sophisticated techniques, VirtualBox virtualizes many guest operating systems entirely in **software**. This means that you can run virtual machines even on older processors which do not support hardware virtualization.

Even though VirtualBox does not always require hardware virtualization, enabling it is *required* in the following scenarios:

- Certain rare guest operating systems like OS/2 make use of very esoteric processor instructions that are not supported with our software virtualization. For virtual machines that are configured to contain such an operating system, hardware virtualization is enabled automatically.

- VirtualBox's 64-bit guest support (added with version 2.0) and multiprocessing (SMP, added with version 3.0) both require hardware virtualization to be enabled. (This is not much of a limitation since the vast majority of today's 64-bit and multicore CPUs ship with hardware virtualization anyway; the exceptions to this rule are e.g. older Intel Celeron and AMD Opteron CPUs.)

Warning: Do not run other hypervisors (open-source or commercial virtualization products) together with VirtualBox! While several hypervisors can normally be *installed* in parallel, do not attempt to *run* several virtual machines from competing hypervisors at the same time. VirtualBox cannot track what another hypervisor is currently attempting to do on the same host, and especially if several products attempt to use hardware virtualization features such as VT-x, this can crash the entire host. Also, within VirtualBox, you can mix software and hardware virtualization when running multiple VMs. In certain cases a small performance penalty will be unavoidable when mixing VT-x and software virtualization VMs. We recommend not mixing virtualization modes if maximum performance and low overhead are essential. This does *not* apply to AMD-V.

10.4 Paravirtualization providers

VirtualBox allows exposing a paravirtualization interface to facilitate accurate and efficient execution of software within a virtual machine. These interfaces require the guest operating system to recognize their presence and make use of them in order to leverage the benefits of communicating with the VirtualBox hypervisor.

Most modern mainstream guest operating systems, including Windows and Linux, ship with support for one or more paravirtualization interfaces. Hence, there is typically no need to install additional software in the guest to take advantage of this feature.

Exposing a paravirtualization provider to the guest operating system does not rely on the choice of host platforms. For example, the *Hyper-V* paravirtualization provider can be used for VMs to run on any host platform (supported by VirtualBox) and not just Windows.

VirtualBox provides the following interfaces:

- **Minimal**: Announces the presence of a virtualized environment. Additionally, reports the TSC and APIC frequency to the guest operating system. This provider is mandatory for running any Mac OS X guests.
- **KVM**: Presents a Linux KVM hypervisor interface which is recognized by Linux kernels starting with version 2.6.25. VirtualBox's implementation currently supports paravirtualized clocks and SMP spinlocks. This provider is recommended for Linux guests.
- **Hyper-V**: Presents a Microsoft Hyper-V hypervisor interface which is recognized by Windows 7 and newer operating systems. VirtualBox's implementation currently supports paravirtualized clocks, APIC frequency reporting, guest debugging, guest crash reporting and relaxed timer checks. This provider is recommended for Windows guests.

10.5 Details about software virtualization

Implementing virtualization on x86 CPUs with no hardware virtualization support is an extraordinarily complex task because the CPU architecture was not designed to be virtualized. The problems can usually be solved, but at the cost of reduced performance. Thus, there is a constant clash between virtualization performance and accuracy.

The x86 instruction set was originally designed in the 1970s and underwent significant changes with the addition of protected mode in the 1980s with the 286 CPU architecture and then again with the Intel 386 and its 32-bit architecture. Whereas the 386 did have limited virtualization support for real mode operation (V86 mode, as used by the "DOS Box" of Windows 3.x and OS/2 2.x), no support was provided for virtualizing the entire architecture.

In theory, software virtualization is not overly complex. In addition to the four privilege levels ("rings") provided by the hardware (of which typically only two are used: ring 0 for kernel mode and ring 3 for user mode), one needs to differentiate between "host context" and "guest context".

In "host context", everything is as if no hypervisor was active. This might be the active mode if another application on your host has been scheduled CPU time; in that case, there is a host ring 3 mode and a host ring 0 mode. The hypervisor is not involved.

In "guest context", however, a virtual machine is active. So long as the guest code is running in ring 3, this is not much of a problem since a hypervisor can set up the page tables properly and run that code natively on the processor. The problems mostly lie in how to intercept what the guest's kernel does.

There are several possible solutions to these problems. One approach is full software emulation, usually involving recompilation. That is, all code to be run by the guest is analyzed, transformed into a form which will not allow the guest to either modify or see the true state of the CPU, and only then executed. This process is obviously highly complex and costly in terms of performance. (VirtualBox contains a recompiler based on QEMU which can be used for pure software emulation, but the recompiler is only activated in special situations, described below.)

Another possible solution is paravirtualization, in which only specially modified guest OSes are allowed to run. This way, most of the hardware access is abstracted and any functions which would normally access the hardware or privileged CPU state are passed on to the hypervisor instead. Paravirtualization can achieve good functionality and performance on standard x86 CPUs, but it can only work if the guest OS can actually be modified, which is obviously not always the case.

VirtualBox chooses a different approach. When starting a virtual machine, through its ring-0 support kernel driver, VirtualBox has set up the host system so that it can run most of the guest code natively, but it has inserted itself at the "bottom" of the picture. It can then assume control when needed – if a privileged instruction is executed, the guest traps (in particular because an I/O register was accessed and a device needs to be virtualized) or external interrupts occur. VirtualBox may then handle this and either route a request to a virtual device or possibly delegate handling such things to the guest or host OS. In guest context, VirtualBox can therefore be in one of three states:

- Guest ring 3 code is run unmodified, at full speed, as much as possible. The number of faults will generally be low (unless the guest allows port I/O from ring 3, something we cannot do as we don't want the guest to be able to access real ports). This is also referred to as "raw mode", as the guest ring-3 code runs unmodified.

- For guest code in ring 0, VirtualBox employs a nasty trick: it actually reconfigures the guest so that its ring-0 code is run in ring 1 instead (which is normally not used in x86 operating systems). As a result, when guest ring-0 code (actually running in ring 1) such as a guest device driver attempts to write to an I/O register or execute a privileged instruction, the VirtualBox hypervisor in "real" ring 0 can take over.

- The hypervisor (VMM) can be active. Every time a fault occurs, VirtualBox looks at the offending instruction and can relegate it to a virtual device or the host OS or the guest OS or run it in the recompiler.

  In particular, the recompiler is used when guest code disables interrupts and VirtualBox cannot figure out when they will be switched back on (in these situations, VirtualBox actually analyzes the guest code using its own disassembler). Also, certain privileged instructions such as LIDT need to be handled specially. Finally, any real-mode or protected-mode code (e.g. BIOS code, a DOS guest, or any operating system startup) is run in the recompiler entirely.

Unfortunately this only works to a degree. Among others, the following situations require special handling:

1. Running ring 0 code in ring 1 causes a lot of additional instruction faults, as ring 1 is not allowed to execute any privileged instructions (of which guest's ring-0 contains plenty). With each of these faults, the VMM must step in and emulate the code to achieve the desired behavior. While this works, emulating thousands of these faults is very expensive and severely hurts the performance of the virtualized guest.

2. There are certain flaws in the implementation of ring 1 in the x86 architecture that were never fixed. Certain instructions that *should* trap in ring 1 don't. This affects, for example, the LGDT/SGDT, LIDT/SIDT, or POPF/PUSHF instruction pairs. Whereas the "load" operation is privileged and can therefore be trapped, the "store" instruction always succeed. If the guest is allowed to execute these, it will see the true state of the CPU, not the virtualized state. The CPUID instruction also has the same problem.

3. A hypervisor typically needs to reserve some portion of the guest's address space (both linear address space and selectors) for its own use. This is not entirely transparent to the guest OS and may cause clashes.

4. The SYSENTER instruction (used for system calls) executed by an application running in a guest OS always transitions to ring 0. But that is where the hypervisor runs, not the guest OS. In this case, the hypervisor must trap and emulate the instruction even when it is not desirable.

5. The CPU segment registers contain a "hidden" descriptor cache which is not software-accessible. The hypervisor cannot read, save, or restore this state, but the guest OS may use it.

6. Some resources must (and can) be trapped by the hypervisor, but the access is so frequent that this creates a significant performance overhead. An example is the TPR (Task Priority) register in 32-bit mode. Accesses to this register must be trapped by the hypervisor, but certain guest operating systems (notably Windows and Solaris) write this register very often, which adversely affects virtualization performance.

To fix these performance and security issues, VirtualBox contains a Code Scanning and Analysis Manager (CSAM), which disassembles guest code, and the Patch Manager (PATM), which can replace it at runtime.

Before executing ring 0 code, CSAM scans it recursively to discover problematic instructions. PATM then performs *in-situ* patching, i.e. it replaces the instruction with a jump to hypervisor memory where an integrated code generator has placed a more suitable implementation. In reality, this is a very complex task as there are lots of odd situations to be discovered and handled correctly. So, with its current complexity, one could argue that PATM is an advanced *in-situ* recompiler.

In addition, every time a fault occurs, VirtualBox analyzes the offending code to determine if it is possible to patch it in order to prevent it from causing more faults in the future. This approach works well in practice and dramatically improves software virtualization performance.

10.6 Details about hardware virtualization

With Intel VT-x, there are two distinct modes of CPU operation: VMX root mode and non-root mode.

- In root mode, the CPU operates much like older generations of processors without VT-x support. There are four privilege levels ("rings"), and the same instruction set is supported, with the addition of several virtualization specific instruction. Root mode is what a host operating system without virtualization uses, and it is also used by a hypervisor when virtualization is active.

- In non-root mode, CPU operation is significantly different. There are still four privilege rings and the same instruction set, but a new structure called VMCS (Virtual Machine Control Structure) now controls the CPU operation and determines how certain instructions behave. Non-root mode is where guest systems run.

Switching from root mode to non-root mode is called "VM entry", the switch back is "VM exit". The VMCS includes a guest and host state area which is saved/restored at VM entry and exit. Most importantly, the VMCS controls which guest operations will cause VM exits.

The VMCS provides fairly fine-grained control over what the guests can and can't do. For example, a hypervisor can allow a guest to write certain bits in shadowed control registers, but not others. This enables efficient virtualization in cases where guests can be allowed to write control bits without disrupting the hypervisor, while preventing them from altering control bits over which the hypervisor needs to retain full control. The VMCS also provides control over interrupt delivery and exceptions.

Whenever an instruction or event causes a VM exit, the VMCS contains information about the exit reason, often with accompanying detail. For example, if a write to the CR0 register causes an exit, the offending instruction is recorded, along with the fact that a write access to a control register caused the exit, and information about source and destination register. Thus the hypervisor can efficiently handle the condition without needing advanced techniques such as CSAM and PATM described above.

VT-x inherently avoids several of the problems which software virtualization faces. The guest has its own completely separate address space not shared with the hypervisor, which eliminates potential clashes. Additionally, guest OS kernel code runs at privilege ring 0 in VMX non-root mode, obviating the problems by running ring 0 code at less privileged levels. For example the SYSENTER instruction can transition to ring 0 without causing problems. Naturally, even at ring 0 in VMX non-root mode, any I/O access by guest code still causes a VM exit, allowing for device emulation.

The biggest difference between VT-x and AMD-V is that AMD-V provides a more complete virtualization environment. VT-x requires the VMX non-root code to run with paging enabled, which precludes hardware virtualization of real-mode code and non-paged protected-mode software. This typically only includes firmware and OS loaders, but nevertheless complicates VT-x hypervisor implementation. AMD-V does not have this restriction.

Of course hardware virtualization is not perfect. Compared to software virtualization, the overhead of VM exits is relatively high. This causes problems for devices whose emulation requires high number of traps. One example is the VGA device in 16-color modes, where not only every I/O port access but also every access to the framebuffer memory must be trapped.

10.7 Nested paging and VPIDs

In addition to "plain" hardware virtualization, your processor may also support additional sophisticated techniques:[2]

- A newer feature called **"nested paging"** implements some memory management in hardware, which can greatly accelerate hardware virtualization since these tasks no longer need to be performed by the virtualization software.

  With nested paging, the hardware provides another level of indirection when translating linear to physical addresses. Page tables function as before, but linear addresses are now translated to "guest physical" addresses first and not physical addresses directly. A new set of paging registers now exists under the traditional paging mechanism and translates from guest physical addresses to host physical addresses, which are used to access memory.

[2] VirtualBox 2.0 added support for AMD's nested paging; support for Intel's EPT and VPIDs was added with version 2.1.

Nested paging eliminates the overhead caused by VM exits and page table accesses. In essence, with nested page tables the guest can handle paging without intervention from the hypervisor. Nested paging thus significantly improves virtualization performance.

On AMD processors, nested paging has been available starting with the Barcelona (K10) architecture – they call it now "rapid virtualization indexing" (RVI). Intel added support for nested paging, which they call "extended page tables" (EPT), with their Core i7 (Nehalem) processors.

If nested paging is enabled, the VirtualBox hypervisor can also use **large pages** to reduce TLB usage and overhead. This can yield a performance improvement of up to 5%. To enable this feature for a VM, you need to use the `VBoxManage modifyvm --largepages` command; see chapter 8.8, *VBoxManage modifyvm*, page 128.

- On Intel CPUs, another hardware feature called **"Virtual Processor Identifiers" (VPIDs)** can greatly accelerate context switching by reducing the need for expensive flushing of the processor's Translation Lookaside Buffers (TLBs).

  To enable these features for a VM, you need to use the `VBoxManage modifyvm --vtxvpid` and `--largepages` commands; see chapter 8.8, *VBoxManage modifyvm*, page 128.

11 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.

The Main API is made available to C++ clients through COM (on Windows hosts) or XPCOM (on other hosts). Bridges also exist for SOAP, Java and Python.

All programming information (documentation, reference information, header and other interface files as well as samples) have been split out to a separate **Software Development Kit (SDK),** which is available for download from http://www.virtualbox.org. In particular, the SDK comes with a "Programming Guide and Reference" in PDF format, which contains, among other things, the information that was previously in this chapter of the User Manual.

12 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.

12.1 Procedures and tools

12.1.1 Categorizing and isolating problems

More often than not, a virtualized guest behaves like a physical system. Any problems that a physical machine would encounter, a virtual machine will encounter as well. If, for example, Internet connectivity is lost due to external issues, virtual machines will be affected just as much as physical ones.

If a true VirtualBox problem is encountered, it helps to categorize and isolate the problem first. Here are some of the questions that should be answered before reporting a problem:

1. Is the problem specific to a certain guest OS? Specific release of a guest OS? Especially with Linux guest related problems, the issue may be specific to a certain distribution and version of Linux.

2. Is the problem specific to a certain host OS? Problems are usually not host OS specific (because most of the VirtualBox code base is shared across all supported platforms), but especially in the areas of networking and USB support, there are significant differences between host platforms. Some GUI related issues are also host specific.

3. Is the problem specific to certain host hardware? This category of issues is typically related to the host CPU. Because of significant differences between VT-x and AMD-V, problems may be specific to one or the other technology. The exact CPU model may also make a difference (even for software virtualization) because different CPUs support different features, which may affect certain aspects of guest CPU operation.

4. Is the problem specific to a certain virtualization mode? Some problems may only occur in software virtualization mode, others may be specific to hardware virtualization.

5. Is the problem specific to guest SMP? That is, is it related to the number of virtual CPUs (VCPUs) in the guest? Using more than one CPU usually significantly affects the internal operation of a guest OS.

6. Is the problem specific to the Guest Additions? In some cases, this is a given (e.g., a shared folders problem), in other cases it may be less obvious (for example, display problems). And if the problem is Guest Additions specific, is it also specific to a certain version of the Additions?

7. Is the problem specific to a certain environment? Some problems are related to a particular environment external to the VM; this usually involves network setup. Certain configurations of external servers such as DHCP or PXE may expose problems which do not occur with other, similar servers.

8. Is the problem a regression? Knowing that an issue is a regression usually makes it significantly easier to find the solution. In this case, it is crucial to know which version is affected and which is not.

12.1.2 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 so-called **"release log file"** is created containing lots of 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 VMs/{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.

The release log file (VBox.log) contains a wealth of diagnostic information, such as Host OS type and version, VirtualBox version and build (32-bit or 64-bit), a complete dump of the guest's configuration (CFGM), detailed information about the host CPU type and supported features, whether hardware virtualization is enabled, information about VT-x/AMD-V setup, state transitions (creating, running, paused, stopping, etc.), guest BIOS messages, Guest Additions messages, device-specific log entries and, at the end of execution, final guest state and condensed statistics.

In case of crashes, it is very important to collect **crash dumps**. This is true for both host and guest crashes. For information about enabling core dumps on Linux, Solaris, and OS X systems, refer to the core dump article on the VirtualBox website.[1]

You can also use `VBoxManage debugvm` to create a dump of a complete virtual machine; see chapter 8.40, *VBoxManage debugvm*, page 177.

For network related problems, it is often helpful to capture a trace of network traffic. If the traffic is routed through an adapter on the host, it is possible to use Wireshark or a similar tool to capture the traffic there. However, this often also includes a lot of traffic unrelated to the VM.

VirtualBox provides an ability to capture network traffic only on a specific VM's network adapter. Refer to the network tracing article on the VirtualBox website[2] for information on enabling this capture. The trace files created by VirtualBox are in `.pcap` format and can be easily analyzed with Wireshark.

12.1.3 The built-in VM debugger

VirtualBox includes a built-in VM debugger, which advanced users may find useful. This debugger allows for examining and, to some extent, controlling the VM state.

> **Warning:** Use the VM debugger at your own risk. There is no support for it, and the following documentation is only made available for advanced users with a very high level of familiarity with the x86/AMD64 machine instruction set, as well as detailed knowledge of the PC architecture. A degree of familiarity with the internals of the guest OS in question may also be very helpful.

The VM debugger is available in all regular production versions of VirtualBox, but it is disabled by default because the average user will have little use for it. There are two ways to access the debugger:

[1] http://www.virtualbox.org/wiki/Core_dump.
[2] http://www.virtualbox.org/wiki/Network_tips.

- A debugger console window displayed alongside the VM
- Via the `telnet` protocol at port 5000

The debugger can be enabled in three ways:

- Start the VM directly using `VirtualBox --startvm`, with an additional `--dbg`, `--debug`, or `--debug-command-line` argument. See the VirtualBox usage help for details.
- Set the `VBOX_GUI_DBG_ENABLED` or `VBOX_GUI_DBG_AUTO_SHOW` environment variable to `true` before launching the VirtualBox process. Setting these variables (only their presence is checked) is effective even when the first VirtualBox process is the VM selector window. VMs subsequently launched from the selector will have the debugger enabled.
- Set the `GUI/Dbg/Enabled` extra data item to `true` before launching the VM. This can be set globally or on a per VM basis.

A new 'Debug' menu entry will be added to the VirtualBox application. This menu allows the user to open the debugger console.

The VM debugger command syntax is loosely modeled on Microsoft and IBM debuggers used on DOS, OS/2 and Windows. Users familiar with symdeb, CodeView, or the OS/2 kernel debugger will find the VirtualBox VM debugger familiar.

The most important command is `help`. This will print brief usage help for all debugger commands. The set of commands supported by the VM debugger changes frequently and the `help` command is always up-to-date.

A brief summary of frequently used commands follows:

- `stop` – stops the VM execution and enables single stepping
- `g` – continue VM execution
- `t` – single step an instruction
- `rg/rh/r` – print the guest/hypervisor/current registers
- `kg/kh/k` – print the guest/hypervisor/current call stack
- `da/db/dw/dd/dq` – print memory contents as ASCII/bytes/words/dwords/qwords
- `u` – unassemble memory
- `dg` – print the guest's GDT
- `di` – print the guest's IDT
- `dl` – print the guest's LDT
- `dt` – print the guest's TSS
- `dp*` – print the guest's page table structures
- `bp/br` – set a normal/recompiler breakpoint
- `bl` – list breakpoints
- `bc` – clear a breakpoint
- `writecore` – writes a VM core file to disk, refer chapter 12.1.4, *VM core format*, page 249

See the built-in `help` for other available commands.

The VM debugger supports symbolic debugging, although symbols for guest code are often not available. For Solaris guests, the `detect` command automatically determines the guest OS version and locates kernel symbols in guest's memory. Symbolic debugging is then available. For Linux guests, the `detect` commands also determines the guest OS version, but there are no symbols in the guest's memory. Kernel symbols are available in the file `/proc/kallsyms` on Linux guests. This file must be copied to the host, for example using `scp`. The `loadmap` debugger command can be used to make the symbol information available to the VM debugger. Note that the `kallsyms` file contains the symbols for the currently loaded modules; if the guest's configuration changes, the symbols will change as well and must be updated.

For all guests, a simple way to verify that the correct symbols are loaded is the `k` command. The guest is normally idling and it should be clear from the symbolic information that the guest operating system's idle loop is being executed.

Another group of debugger commands is the set of `info` commands. Running `info help` provides complete usage information. The information commands provide ad-hoc data pertinent to various emulated devices and aspects of the VMM. There is no general guideline for using the `info` commands, the right command to use depends entirely on the problem being investigated. Some of the info commands are:

- `cfgm` – print a branch of the configuration tree
- `cpuid` – display the guest CPUID leaves
- `ioport` – print registered I/O port ranges
- `mmio` – print registered MMIO ranges
- `mode` – print the current paging mode
- `pit` – print the i8254 PIT state
- `pic` – print the i8259A PIC state
- `ohci/ehci/xhci` – print a subset of the OHCI/EHCI/xHCI USB controller state
- `pcnet0` – print the PCnet state
- `vgatext` – print the contents of the VGA framebuffer formatted as standard text mode
- `timers` – print all VM timers

The output of the `info` commands generally requires in-depth knowledge of the emulated device and/or VirtualBox VMM internals. However, when used properly, the information provided can be invaluable.

12.1.4 VM core format

VirtualBox uses the 64-bit ELF format for its VM core files created by `VBoxManage debugvm`; see chapter 8.40, *VBoxManage debugvm*, page 177. The VM core file contain the memory and CPU dumps of the VM and can be useful for debugging your guest OS. The 64-bit ELF object format specification can be obtained here: `http://downloads.openwatcom.org/ftp/devel/docs/elf-64-gen.pdf`.

The overall layout of the VM core format is as follows:

```
[ ELF 64 Header]
[ Program Header, type PT_NOTE ]
  → offset to COREDESCRIPTOR
[ Program Header, type PT_LOAD ] - one for each contiguous physical memory range
  → Memory offset of range
```

```
  → File offset
[ Note Header, type NT_VBOXCORE ]
[ COREDESCRIPTOR ]
  → Magic
  → VM core file version
  → VBox version
  → Number of vCPUs etc.
[ Note Header, type NT_VBOXCPU ] - one for each vCPU
[ vCPU 1 Note Header ]
  [ DBGFCORECPU - vCPU 1 dump ]
[ Additional Notes + Data ] - currently unused
[ Memory dump ]
```

The memory descriptors contain physical addresses relative to the guest and not virtual addresses. Regions of memory such as MMIO regions are not included in the core file.

The relevant data structures and definitions can be found in the VirtualBox sources under the following header files: `include/VBox/dbgfcorefmt.h`, `include/iprt/x86.h` and `src/VBox/Runtime/include/internal/ldrELFCommon.h`.

The VM core file can be inspected using `elfdump` and GNU `readelf` or other similar utilities.

12.2 General

12.2.1 Guest shows IDE/SATA errors for file-based images 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/SATA 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 an image 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 image file 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.

For IDE disks use the following command:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/piix3ide/0/LUN#[x]/Config/FlushInterval" [b]
```

For SATA disks use the following command:

```
VBoxManage setextradata "VM name"
      "VBoxInternal/Devices/ahci/0/LUN#[x]/Config/FlushInterval" [b]
```

The value [x] that selects the disk for IDE 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. For SATA use values between 0 and 29. Only disks support this configuration option; it must not be set for CD/DVD 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.

12.2.2 Responding to guest IDE/SATA flush requests

If desired, the virtual disk images can be flushed when the guest issues the IDE FLUSH CACHE command. Normally these requests are ignored for improved performance. The parameters below are only accepted for disk drives. They must not be set for DVD drives.

To enable flushing for IDE disks, issue the following command:

```
VBoxManage setextradata "VM name" "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.

To enable flushing for SATA disks, issue the following command:

```
VBoxManage setextradata "VM name" "VBoxInternal/Devices/ahci/0/LUN#[x]/Config/IgnoreFlush" 0
```

The value [x] that selects the disk can be a value between 0 and 29.

Note that this doesn't affect the flushes performed according to the configuration described in chapter 12.2.1, *Guest shows IDE/SATA errors for file-based images on slow host file system*, page 250. Restoring the default of ignoring flush commands is possible by setting the value to 1 or by removing the key.

12.2.3 Performance variation with frequency boosting

Many newer multi-core processors support some form of frequency boosting, which means that if only one core is utilized, it can run faster (possibly 50% faster or even more) than the rated CPU frequency. This causes measured performance to vary somewhat as a function of the momentary overall system load. The exact behavior depends strongly on the specific processor model.

As a consequence, benchmarking on systems which utilize frequency boosting may produce unstable and non-repeatable results, especially if benchmark runs are short (on the order of seconds). To obtain stable results, benchmarks must be run over longer periods of time and with a constant system load apart from the VM being tested.

12.2.4 Frequency scaling effect on CPU usage

On some hardware platforms and operating systems, CPU frequency scaling may cause CPU usage reporting to be highly misleading. This happens in situations when the host CPU load is significant but not heavy, such as 15-30% of the maximum.

Most operating systems determine CPU usage in terms of time spent, measuring for example how many nanoseconds the systems or a process was active within one second. However, in order to save energy, modern systems can significantly scale down CPU speed when the system is not fully loaded. Naturally, when the CPU is running at (for example) one half of its maximum speed, the same number of instructions will take roughly twice as long to execute compared to running at full speed.

Depending on the specific hardware and host OS, this effect can very significantly skew the CPU usage reported by the OS; the reported CPU usage can be several times higher than what it would have been had the CPU been running at full speed. The effect can be observed both on the host OS and in a guest OS.

12.2.5 Inaccurate Windows CPU usage reporting

CPU usage reporting tools which come with Windows (Task Manager, Resource Monitor) do not take the time spent processing hardware interrupts into account. If the interrupt load is heavy (thousands of interrupts per second), CPU usage may be significantly underreported.

This problem affects Windows as both host and guest OS. Sysinternals tools (e.g. Process Explorer) do not suffer from this problem.

12.2.6 Poor performance caused by host power management

On some hardware platforms and operating systems, virtualization performance is negatively affected by host CPU power management. The symptoms may be choppy audio in the guest or erratic guest clock behavior.

Some of the problems may be caused by firmware and/or host operating system bugs. Therefore, updating the firmware and applying operating systems fixes is recommended.

For optimal virtualization performance, the C1E power state support in the system's BIOS should be disabled, if such a setting is available (not all systems support the C1E power state). On Intel systems the `Intel C State` setting should be disabled. Disabling other power management settings may also improve performance. However, a balance between performance and power consumption must always be considered.

12.2.7 GUI: 2D Video Acceleration option is grayed out

To use 2D Video Acceleration within VirtualBox, your host's video card should support certain OpenGL extensions. On startup, VirtualBox checks for those extensions, and, if the test fails, this option is silently grayed out.

To find out why it has failed, you can manually execute the following command:

```
VBoxTestOGL --log "log_file_name" --test 2D
```

It will list the required OpenGL extensions one by one and will show you which one failed the test. This usually means that you are running an outdated or misconfigured OpenGL driver on your host. It can also mean that your video chip is lacking required functionality.

12.3 Windows guests

12.3.1 No USB 3.0 support in Windows 7 guests

If a Windows 7 or Windows Server 2008 R2 guest is configured for USB 3.0 (xHCI) support, the guest OS will not have any USB support at all. This happens because Windows 7 predates USB 3.0 and therefore does not ship with any xHCI drivers; Microsoft also does not offer any vendor-provided xHCI drivers via Windows Update.

To solve this problem, it is necessary to download and install the Intel xHCI driver in the guest. Intel offers the driver as the USB 3.0 eXtensible Host Controller (xHCI) driver for Intel 7 Series/C216 chipsets.

Note that the driver only supports Windows 7 and Windows Server 2008 R2. The driver package includes support for both 32-bit and 64-bit OS variants.

12.3.2 Windows bluescreens after changing VM configuration

Changing certain virtual machine settings can cause Windows guests to fail during start up with a bluescreen. This may happen if you change VM settings after installing Windows, or if you copy a disk image with an already installed Windows to a newly created VM which has settings that differ from the original machine.

This applies in particular to the following settings:

- The ACPI and I/O APIC settings should never be changed after installing Windows. Depending on the presence of these hardware features, the Windows installation program chooses special kernel and device driver versions and will fail to startup should these hardware features be removed. (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.)
- Changing the storage controller hardware will cause bootup failures as well. This might also apply to you if you copy a disk image from an older version of VirtualBox to a virtual machine created with a newer VirtualBox version; the default subtype of IDE controller hardware was changed from PIIX3 to PIIX4 with VirtualBox 2.2. Make sure these settings are identical.

12.3.3 Windows 0x101 bluescreens with SMP enabled (IPI timeout)

If a VM is configured to have more than one processor (symmetrical multiprocessing, SMP), some configurations of Windows guests crash with an 0x101 error message, indicating a timeout for inter-processor interrupts (IPIs). These interrupts synchronize memory management between processors.

According to Microsoft, this is due to a race condition in Windows. A hotfix is available.[3] If this does not help, please reduce the number of virtual processors to 1.

12.3.4 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 "VM name" "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).

12.3.5 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:

[3] See http://support.microsoft.com/kb/955076.

```
VBoxManage setextradata "VM name" "VBoxInternal/PDM/HaltOnReset" 1
```

12.3.6 PCnet driver failure in 32-bit Windows Server 2003 guests

Certain editions of Windows 2000 and 2003 servers support more than 4 GB RAM on 32-bit systems. The AMD PCnet network driver shipped with Windows Server 2003 fails to load if the 32-bit guest OS uses paging extensions (which will occur with more than approximately 3.5 GB RAM assigned to the VM).

This problem is known to occur with version 4.38.0.0 of the PCnet driver. The issue was fixed in version 4.51.0.0 of the driver, which is available as a separate download. An alternative solution may be changing the emulated NIC type to Intel PRO/1000 MT Desktop (82540EM), or reducing the RAM assigned to the VM to approximately 3.5 GB or less.

12.3.7 No networking in Windows Vista guests

With Windows Vista, Microsoft dropped support for the AMD PCNet card that VirtualBox used to provide as the default virtual network card before version 1.6.0. For Windows Vista guests, VirtualBox now uses an Intel E1000 card by default.

If, for some reason, you still want to use the AMD card, you need to download the PCNet driver from the AMD website (available for 32-bit Windows only). You can transfer it into the virtual machine using a shared folder, see (see chapter 4.3, *Shared folders*, page 68).

12.3.8 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.

12.3.9 Long delays when accessing shared folders

The performance for accesses to shared folders from a Windows guest might be decreased due to delays during the resolution of the VirtualBox shared folders name service. To fix these delays, add the following entries to the file `\windows\system32\drivers\etc\lmhosts` of the Windows guest:

```
255.255.255.255        VBOXSVR #PRE
255.255.255.255        VBOXSRV #PRE
```

After doing this change, a reboot of the guest is required.

12.3.10 USB tablet coordinates wrong in Windows 98 guests

If a Windows 98 VM is configured to use the emulated USB tablet (absolute pointing device), the coordinate translation may be incorrect and the pointer is restricted to the upper left quarter of the guest's screen.

The USB HID (Human Interface Device) drivers in Windows 98 are very old and do not handle tablets the same way all more recent operating systems do (Windows 2000 and later, Mac OS X, Solaris). To work around the problem, issue the following command:

```
VBoxManage setextradata "VM name" "VBoxInternal/USB/HidMouse/0/Config/CoordShift" 0
```

To restore the default behavior, remove the key or set its value to 1.

12.3.11 Windows guests are removed from an Active Directory domain after restoring a snapshot

If a Windows guest is a member of an Active Directory domain and the snapshot feature of VirtualBox is used, it could happen it loses this status after you restore an older snapshot.

The reason is the automatic machine password changing performed by Windows in regular intervals for security purposes. You can disable this feature by following the instruction of this http://support.microsoft.com/kb/154501 article from Microsoft.

12.3.12 Restoring d3d8.dll and d3d9.dll

VirtualBox Guest Additions for Windows prior to 4.1.8 did not properly back up the original d3d8.dll and d3d9.dll system files when selecting and installing the experimental Direct3D support. This process replaces both system files with files from the VirtualBox Guest Additions so that Direct3D calls can be handled correctly. Although this issue was fixed with VirtualBox 4.1.8, there is no way the Windows Guest Additions installer can repair these files.

Corruption of these files has no implications in case 3D acceleration is enabled and basic Direct3D support is installed, that is, without WDDM (on Windows Vista or higher) or on older Windows systems like Windows XP. With the basic Direct3D support all Direct3D 8.0 and Direct3D 9.0 applications will utilize VirtualBox Direct3D files directly and thus will run as expected.

For WDDM Direct3D support however, the originally shipped d3d8.dll and d3d9.dll files are required in order to run Direct3D 8.0 and Direct3D 9.0 applications. As a result of the above mentioned system files corruption these applications will not work anymore. See below for a step-by-step guide for restoring the original d3d8.dll and d3d9.dll system files in case the VirtualBox Guest Additions installer warned about those incorrect files or when having trouble running Direct3D applications.

> **Note:** Starting at Windows 7 the 3D desktop (aka Aero) uses DirectX 10 for rendering so that corrupted d3d8.dll and d3d9.dll system files will have no effect on the actual rendering.

This is why such a detected file corruption is not considered as fatal for the basic Direct3D installation on all supported Windows guests, and for WDDM Direct3D installation on Windows 7 and later guests.

Extracting d3d8 and d3d9.dll from a Windows XP installation CD:

1. Download and install the latest version of 7-Zip File Manager http//www.7-zip.org
2. Browse into the installation CD for example E:\i386 (or amd64 for the 64-bit version)
3. Locate file d3d8.dl_ and d3d9.dl_, double click on it and Extract d3d8.dll and d3d9.dll
4. Reboot Windows in Safe mode
5. Copy extracted d3d8.dll and d3d9.dll to C:\Windows\system32 and C:\Windows\system32\dllcache
6. Reboot

Extracting d3d8 and d3d9.dll from Windows XP Service pack

1. 1, 3-6 Same as installation CD
2. Use 'Open inside' to open WindowsXP-KB936929-SP3-x86.exe as archive and browse i386 directory.

Extracting d3d8 and d3d9.dll from Vista/Windows7 installation CD or Service Pack iso

1. Download and install the latest version of 7-Zip File Manager http//www.7-zip.org
2. Browse into installation CD for example E:\sources
3. Locate file install.wim and double click it. After 7-Zip utility opens the file, you'll get a few numbered folders. Each numeric subfolder represents a different version of Windows (Starter, Home Basic, and so on)
4. After entering into the one of the numeric folders, browse into Windows\System32 (or C:\Windows\SysWOW64 for the 64-bit version) directory locate d3d8.dll and d3d9.dll and extract
5. Copy extracted d3d8.dll and d3d9.dll to C:\Windows\system32 or C:\Windows\SysWOW64 (files from system32 should go to system32, from SysWOW64 to SysWOW64)
6. Reboot

12.3.13 Windows 3.x limited to 64 MB RAM

Windows 3.x guests are typically limited to 64 MB RAM, even if a VM is assigned much more memory. While Windows 3.1 is theoretically capable of using up to 512 MB RAM, it only uses memory available through the XMS interface. Versions of HIMEM.SYS (the Microsoft XMS manager) shipped with MS-DOS and Microsoft Windows 3.x can only use up to 64 MB on standard PCs.

This is a HIMEM.SYS limitation documented by Microsoft in Knowledge base article KB 116256. Windows 3.1 memory limits are described in detail in Microsoft Knowledge base article KB 84388.

It is possible for Windows 3.x guests to utilize more than 64 MB RAM if a different XMS provider is used. That could be a newer HIMEM.SYS version (such as that shipped with Windows 98), or a more capable third-party memory manager (such as QEMM).

12.4 Linux and X11 guests

12.4.1 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.

Linux kernels shipped with Red Hat Enterprise Linux (RHEL) as of release 4.7 and 5.1 as well as kernels of related Linux distributions (for instance CentOS and Oracle Linux) support a kernel parameter *divider=N*. Hence, such kernels support a lower timer frequency without recompilation. We suggest to add the kernel parameter *divider=10* to select a guest kernel timer frequency of 100Hz.

12.4.2 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 I/O-APIC to work around the problem (see chapter 3.4, *System settings*, page 49).

12.4.3 Buggy Linux 2.6 kernel versions

The following bugs in Linux kernels prevent them from executing correctly in VirtualBox, causing VM boot crashes:

- The Linux kernel version 2.6.18 (and some 2.6.17 versions) introduced a race condition that can cause boot crashes in VirtualBox. Please use a kernel version 2.6.19 or later.
- With hardware virtualization and the I/O APIC enabled, kernels before 2.6.24-rc6 may panic on boot with the following message:

  ```
  Kernel panic - not syncing: IO-APIC + timer doesn't work!  Boot with
  apic=debug and send a report.  Then try booting with the 'noapic' option
  ```

  If you see this message, either disable hardware virtualization or the I/O APIC (see chapter 3.4, *System settings*, page 49), or upgrade the guest to a newer kernel.[4]

12.4.4 Shared clipboard, auto-resizing and seamless desktop in X11 guests

Guest desktop services in guests running the X11 window system (Solaris, Linux and others) are provided by a guest service called `VBoxClient`, which runs under the ID of the user who started the desktop session and is automatically started using the following command lines

```
VBoxClient --clipboard
VBoxClient --display
VBoxClient --seamless
```

when your X11 user session is started if you are using a common desktop environment (Gnome, KDE and others). If a particular desktop service is not working correctly, it is worth checking whether the process which should provide it is running.

The `VBoxClient` processes create files in the user's home directory with names of the form `.vboxclient-*.pid` when they are running in order to prevent a given service from being started twice. It can happen due to misconfiguration that these files are created owned by root and not deleted when the services are stopped, which will prevent them from being started in future sessions. If the services cannot be started, you may wish to check whether these files still exist.

12.5 Solaris guests

12.5.1 Older Solaris 10 releases crash in 64-bit mode

Solaris 10 releases up to and including Solaris 10 8/07 ("S10U4") incorrectly detect newer Intel processors produced since 2007. This problem leads to the 64-bit Solaris kernel crashing (and usually causing a triple fault) almost immediately during startup, in both virtualized and physical environments.

The recommended solution is upgrading to at least Solaris 10 5/08 ("S10U5"). Alternative solutions include forcing Solaris to always boot the 32-bit kernel or applying a patch for bug 6574102 (while Solaris is using the 32-bit kernel).

[4] See http://www.mail-archive.com/git-commits-head@vger.kernel.org/msg30813.html for details about the kernel fix.

12.5.2 Certain Solaris 10 releases may take long to boot with SMP

When using more than one CPU, Solaris 10 releases 5/08 ("S10U5"), 10/08 ("S10U6"), and 5/09 ("S10U7") may take a long time to boot and may print warnings on the system console regarding failures to read from disk. This is a bug in Solaris 10 which affects specific physical and virtual configurations. It is caused by trying to read microcode updates from the boot disk when the disk interrupt is reassigned to a not yet fully initialized secondary CPU. Disk reads will time out and fail, triggering delays (approx. 45 seconds) and warnings.

The recommended solution is upgrading to at least Solaris 10 10/09 ("S10U8") which includes a fix for this problem. Alternative solutions include restricting the number of virtual CPUs to one or possibly using a different storage controller.

12.5.3 Solaris 8 5/01 and earlier may crash on startup

Solaris 2.6, 7 and 8 releases up to and including Solaris 8 4/01 ("S8U4") incorrectly set up Machine Check Exception (MCE) MSRs on Pentium 4 and some later Intel CPUs. The problem leads to the Solaris kernel crashing (and usually causing a triple fault) almost immediately during startup, in both virtualized and physical environments. Solaris 9 and later releases are not affected by this problem, and neither is Solaris 2.5.1 and earlier.

The recommended solution is upgrading to at least Solaris 8 7/01 ("S8U5"). Alternative solutions include applying a patch for bugs 4408508 and 4414557 (on an unaffected system).

12.6 FreeBSD guests

12.6.1 FreeBSD 10.0 may hang with xHCI

If xHCI (USB 3.0) emulation is enabled for FreeBSD 10.0 guests, the guest OS will hang. This is caused by the guest OS incorrectly handling systems where MSIs (Message Signaled Interrupts) are not used with the xHCI device.

The problem does not exist in earlier FreeBSD releases and was fixed in FreeBSD 10.1.

12.7 Windows hosts

12.7.1 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`.

12.7.2 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.

12.7.3 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 RDP server.

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.

12.7.4 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.

12.7.5 Bridged networking adapters missing

If no bridged adapters show up in the "Networking" section of the VM settings, this typically means that the bridged networking driver was not installed properly on your host. This could be due to the following reasons:

- The maximum allowed filter count was reached on the host. In this case, the MSI log would mention the `0x8004a029` error code returned on NetFlt network component install:

  ```
  VBoxNetCfgWinInstallComponent: Install failed, hr (0x8004a029)
  ```

  You can try to increase the maximum filter count in the Windows registry at the following key:

  ```
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Network\MaxNumFilters
  ```

  The maximum number allowed is 14. After a reboot, try to re-install VirtualBox.

- The INF cache is corrupt. In this case, the install log (`%windir%\inf\setupapi.log` on XP or `%windir%\inf\setupapi.dev.log` on Vista or later) would typically mention the failure to find a suitable driver package for either the `sun_VBoxNetFlt` or `sun_VBoxNetFltmp` components. The solution then is to uninstall VirtualBox, remove the INF cache (`%windir%\inf\INFCACHE.1`), reboot and try to re-install VirtualBox

12.7.6 Host-only networking adapters cannot be created

If host-only adapter cannot be created (either via the Manager or VBoxManage), then the INF cache is probably corrupt. In this case, the install log (`%windir%\inf\setupapi.log` on XP or `%windir%\inf\setupapi.dev.log` on Vista or later) would typically mention the failure to find a suitable driver package for the `sun_VBoxNetAdp` component. Again, as with the bridged networking problem described above, the solution is to uninstall VirtualBox, remove the INF cache (`%windir%\inf\INFCACHE.1`), reboot and try to re-install VirtualBox.

12.8 Linux hosts

12.8.1 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. Most probably the kernel disagrees with the version of the gcc used to compile the module. Make sure that you use the same compiler as used to build the kernel.

12.8.2 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`).

12.8.3 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.

12.8.4 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.

12.8.5 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.

12.8.6 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 `VBoxXPCOMIPCD` 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 `~/.config/VirtualBox`, or the appropriate configuration directory for your

operating system. 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.

12.8.7 USB not working

If USB is not working on your Linux host, make sure that the current user is a member of the `vboxusers` group. Please keep in mind that group membership does not take effect immediately but rather at the next login. If available, the `newgrp` command may avoid the need for logout/login.

12.8.8 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.

12.8.9 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 1 GB 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 have 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
```

12.9 Solaris hosts

12.9.1 Cannot start VM, not enough contiguous memory

The ZFS file system is known to use nearly all available RAM as cache if the default system settings are not changed. This may lead to a heavy fragmentation of the host memory preventing VirtualBox VMs from being started. We recommend to limit the ZFS cache by adding a line

```
set zfs:zfs_arc_max = xxxx
```

to /etc/system where `xxxx` bytes is the amount of memory usable for the ZFS cache.

12.9.2 VM aborts with out of memory errors on Solaris 10 hosts

32-bit Solaris 10 hosts (bug 1225025) require swap space equal to, or greater than the host's physical memory size. For example, 8 GB physical memory would require at least 8 GB swap. This can be configured during a Solaris 10 install by choosing a 'custom install' and changing the default partitions.

Note: This restriction applies only to 32-bit Solaris hosts, 64-bit hosts are not affected!

For existing Solaris 10 installs, an additional swap image needs to be mounted and used as swap. Hence if you have 1 GB swap and 8 GB of physical memory, you require to add 7 GB more swap. This can be done as follows:

For ZFS (as root user):

```
zfs create -V 8gb /_<ZFS volume>_/swap
swap -a /dev/zvol/dsk/_<ZFS volume>_/swap
```

To mount if after reboot, add the following line to /etc/vfstab:

```
/dev/zvol/dsk/_<ZFS volume>_/swap - - swap - no -
```

Alternatively, you could grow the existing swap using:

```
zfs set volsize=8G rpool/swap
```

And reboot the system for the changes to take effect.

For UFS (as root user):

```
mkfile 7g /path/to/swapfile.img
swap -a /path/to/swapfile.img
```

To mount it after reboot, add the following line to /etc/vfstab:

```
/path/to/swap.img - - swap - no -
```

13 Security guide

13.1 General Security Principles

The following principles are fundamental to using any application securely.

Keep Software Up To Date One of the principles of good security practise is to keep all software versions and patches up to date. Activate the VirtualBox update notification to get notified when a new VirtualBox release is available. When updating VirtualBox, do not forget to update the Guest Additions. Keep the host operating system as well as the guest operating system up to date.

Restrict Network Access to Critical Services Use proper means, for instance a firewall, to protect your computer and your guest(s) from accesses from the outside. Choosing the proper networking mode for VMs helps to separate host networking from the guest and vice versa.

Follow the Principle of Least Privilege The principle of least privilege states that users should be given the least amount of privilege necessary to perform their jobs. Always execute VirtualBox as a regular user. We strongly discourage anyone from executing VirtualBox with system privileges.

Choose restrictive permissions when creating configuration files, for instance when creating /etc/default/virtualbox, see chapter 2.3.3.7, *Automatic installation options*, page 41. Mode 0600 would be preferred.

Monitor System Activity System security builds on three pillars: good security protocols, proper system configuration and system monitoring. Auditing and reviewing audit records address the third requirement. Each component within a system has some degree of monitoring capability. Follow audit advice in this document and regularly monitor audit records.

Keep Up To Date on Latest Security Information Oracle continually improves its software and documentation. Check this note yearly for revisions.

13.2 Secure Installation and Configuration

13.2.1 Installation Overview

The VirtualBox base package should be downloaded only from a trusted source, for instance the official website `http://www.virtualbox.org`. The integrity of the package should be verified with the provided SHA256 checksum which can be found on the official website.

General VirtualBox installation instructions for the supported hosts can be found in chapter 2, *Installation details*, page 34.

On Windows hosts, the installer allows for disabling USB support, support for bridged networking, support for host-only networking and the Python language bindings, see chapter 2.1, *Installing on Windows hosts*, page 34. All these features are enabled by default but disabling some of them could be appropriate if the corresponding functionality is not required by any virtual machine. The Python language bindings are only required if the VirtualBox API is to be used by external Python applications. In particular USB support and support for the two networking modes require the installation of Windows kernel drivers on the host. Therefore disabling those

selected features can not only be used to restrict the user to certain functionality but also to minimize the surface provided to a potential attacker.

The general case is to install the complete VirtualBox package. The installation must be done with system privileges. All VirtualBox binaries should be executed as a regular user and never as a privileged user.

The Oracle VM VirtualBox extension pack provides additional features and must be downloaded and installed separately, see chapter 1.5, *Installing VirtualBox and extension packs*, page 16. As for the base package, the SHA256 checksum of the extension pack should be verified. As the installation requires system privileges, VirtualBox will ask for the system password during the installation of the extension pack.

13.2.2 Post Installation Configuration

Normally there is no post installation configuration of VirtualBox components required. However, on Solaris and Linux hosts it is necessary to configure the proper permissions for users executing VMs and who should be able to access certain host resources. For instance, Linux users must be member of the *vboxusers* group to be able to pass USB devices to a guest. If a serial host interface should be accessed from a VM, the proper permissions must be granted to the user to be able to access that device. The same applies to other resources like raw partitions, DVD/CD drives and sound devices.

13.3 Security Features

This section outlines the specific security mechanisms offered by VirtualBox.

13.3.1 The Security Model

One property of virtual machine monitors (VMMs) like VirtualBox is to encapsulate a guest by executing it in a protected environment, a virtual machine, running as a user process on the host operating system. The guest cannot communicate directly with the hardware or other computers but only through the VMM. The VMM provides emulated physical resources and devices to the guest which are accessed by the guest operating system to perform the required tasks. The VM settings control the resources provided to the guest, for example the amount of guest memory or the number of guest processors, (see chapter 3.3, *General settings*, page 48) and the enabled features for that guest (for example remote control, certain screen settings and others).

13.3.2 Secure Configuration of Virtual Machines

Several aspects of a virtual machine configuration are subject to security considerations.

13.3.2.1 Networking

The default networking mode for VMs is NAT which means that the VM acts like a computer behind a router, see chapter 6.3, *Network Address Translation (NAT)*, page 94. The guest is part of a private subnet belonging to this VM and the guest IP is not visible from the outside. This networking mode works without any additional setup and is sufficient for many purposes.

If bridged networking is used, the VM acts like a computer inside the same network as the host, see chapter 6.5, *Bridged networking*, page 97. In this case, the guest has the same network access as the host and a firewall might be necessary to protect other computers on the subnet from a potential malicious guest as well as to protect the guest from a direct access from other computers. In some cases it is worth considering using a forwarding rule for a specific port in NAT mode instead of using bridged networking.

Some setups do not require a VM to be connected to the public network at all. Internal networking (see chapter 6.6, *Internal networking*, page 98) or host-only networking (see chapter 6.7, *Host-only networking*, page 99) are often sufficient to connect VMs among each other or to connect VMs only with the host but not with the public network.

13.3.2.2 VRDP remote desktop authentication

When using the VirtualBox extension pack provided by Oracle for VRDP remote desktop support, you can optionally use various methods to configure RDP authentication. The "null" method is very insecure and should be avoided in a public network. See chapter 7.1.5, *RDP authentication*, page 108 for details.

13.3.2.3 Clipboard

The shared clipboard allows users to share data between the host and the guest. Enabling the clipboard in "Bidirectional" mode allows the guest to read and write the host clipboard. The "Host to guest" mode and the "Guest to host" mode limit the access to one direction. If the guest is able to access the host clipboard it can also potentially access sensitive data from the host which is shared over the clipboard.

If the guest is able to read from and/or write to the host clipboard then a remote user connecting to the guest over the network will also gain this ability, which may not be desirable. As a consequence, the shared clipboard is disabled for new machines.

13.3.2.4 Shared folders

If any host folder is shared with the guest then a remote user connected to the guest over the network can access these files too as the folder sharing mechanism cannot be selectively disabled for remote users.

13.3.2.5 3D graphics acceleration

Enabling 3D graphics via the Guest Additions exposes the host to additional security risks; see chapter 4.5.1, *Hardware 3D acceleration (OpenGL and Direct3D 8/9)*, page 72.

13.3.2.6 CD/DVD passthrough

Enabling CD/DVD passthrough allows the guest to perform advanced operations on the CD/DVD drive, see chapter 5.9, *CD/DVD support*, page 90. This could induce a security risk as a guest could overwrite data on a CD/DVD medium.

13.3.2.7 USB passthrough

Passing USB devices to the guest provides the guest full access to these devices, see chapter 3.10.1, *USB settings*, page 56. For instance, in addition to reading and writing the content of the partitions of an external USB disk the guest will be also able to read and write the partition table and hardware data of that disk.

13.3.3 Configuring and Using Authentication

The following components of VirtualBox can use passwords for authentication:

- When using remote iSCSI storage and the storage server requires authentication, an initiator secret can optionally be supplied with the `VBoxManage storageattach` command. As long as no settings password is provided (command line option

`--settingspwfile`

, this secret is stored **unencrypted** in the machine configuration and is therefore potentially readable on the host. See chapter 5.10, *iSCSI servers*, page 91 and chapter 8.18, *VBoxManage storageattach*, page 150.

- When using the VirtualBox web service to control a VirtualBox host remotely, connections to the web service are authenticated in various ways. This is described in detail in the VirtualBox Software Development Kit (SDK) reference; please see chapter 11, *VirtualBox programming interfaces*, page 245.

13.3.4 Potentially insecure operations

The following features of VirtualBox can present security problems:

- Enabling 3D graphics via the Guest Additions exposes the host to additional security risks; see chapter 4.5.1, *Hardware 3D acceleration (OpenGL and Direct3D 8/9)*, page 72.
- When teleporting a machine, the data stream through which the machine's memory contents are transferred from one host to another is not encrypted. A third party with access to the network through which the data is transferred could therefore intercept that data. An SSH tunnel could be used to secure the connection between the two hosts. But when considering teleporting a VM over an untrusted network the first question to answer is how both VMs can securely access the same virtual disk image(s) with a reasonable performance.
- When Page Fusion (see chapter 4.9.2, *Page Fusion*, page 77) is enabled, it is possible that a side-channel opens up that allows a malicious guest to determine the address space layout (i.e. where DLLs are typically loaded) of one other VM running on the same host. This information leak in it self is harmless, however the malicious guest may use it to optimize attack against that VM via unrelated attack vectors. It is recommended to only enable Page Fusion if you do not think this is a concern in your setup.
- When using the VirtualBox web service to control a VirtualBox host remotely, connections to the web service (through which the API calls are transferred via SOAP XML) are not encrypted, but use plain HTTP by default. This is a potential security risk! For details about the web service, please see chapter 11, *VirtualBox programming interfaces*, page 245.

  The web services are not started by default. Please refer to chapter 9.21, *Starting the VirtualBox web service automatically*, page 218 to find out how to start this service and how to enable SSL/TLS support. It has to be started as a regular user and only the VMs of that user can be controlled. By default, the service binds to localhost preventing any remote connection.
- Traffic sent over a UDP Tunnel network attachment is not encrypted. You can either encrypt it on the host network level (with IPsec), or use encrypted protocols in the guest network (such as SSH). The security properties are similar to bridged Ethernet.
- Because of shortcomings in older Windows versions, using VirtualBox on Windows versions older than Vista with Service Pack 1 is not recommended.

13.3.5 Encryption

The following components of VirtualBox use encryption to protect sensitive data:

- When using the VirtualBox extension pack provided by Oracle for VRDP remote desktop support, RDP data can optionally be encrypted. See chapter 7.1.6, *RDP encryption*, page 109 for details. Only the Enhanced RDP Security method (RDP5.2) with TLS protocol provides a secure connection. Standard RDP Security (RDP4 and RDP5.1) is vulnerable to a man-in-the-middle attack.

14 Known limitations

14.1 Experimental Features

Some VirtualBox features are labeled as experimental. Such features are provided on an "as-is" basis and are not formally supported. However, feedback and suggestions about such features are welcome. A comprehensive list of experimental features follows:

- WDDM Direct3D video driver for Windows guests
- Hardware 3D acceleration support for Windows, Linux, and Solaris guests
- Hardware 2D video playback acceleration support for Windows guests
- PCI pass-through (Linux hosts only)
- Mac OS X guests (Mac hosts only)
- ICH9 chipset emulation
- EFI firmware
- Host CD/DVD drive pass-through
- Support of iSCSI via internal networking

14.2 Known Issues

The following section describes known problems with VirtualBox 5.1.22. Unless marked otherwise, these issues are planned to be fixed in later releases.

- The following **Guest SMP (multiprocessor) limitations** exist:
  - **Poor performance** with 32-bit guests on AMD CPUs. This affects mainly Windows and Solaris guests, but possibly also some Linux kernel revisions. Partially solved in 3.0.6 for 32 bits Windows NT, 2000, XP and 2003 guests. Requires 3.0.6 or higher Guest Additions to be installed.
  - **Poor performance** with 32-bit guests on certain Intel CPU models that do not include virtual APIC hardware optimization support. This affects mainly Windows and Solaris guests, but possibly also some Linux kernel revisions. Partially solved in 3.0.12 for 32 bits Windows NT, 2000, XP and 2003 guests. Requires 3.0.12 or higher Guest Additions to be installed.
- **NX (no execute, data execution prevention)** only works for guests running on 64-bit hosts or guests running on 32-bit hosts with PAE enabled and requires that hardware virtualization be enabled.
- For **basic Direct3D support in Windows guests** to work, the Guest Additions must be installed in Windows "safe mode". Press F8 when the Windows guest is booting and select "Safe mode", then install the Guest Additions. Otherwise Windows' file protection mechanism will interfere with the replacement DLLs installed by VirtualBox and keep restoring the original Windows system DLLs.

> **Note:** This does **not** apply to the experimental WDDM Direct3D video driver available for Vista and Windows 7 guests shipped with VirtualBox 4.1.

- **Guest control.** On Windows guests, a process lauched via the guest control execute support will not be able to display a graphical user interface *unless* the user account under which it is running is currently logged in and has a desktop session.

  Also, to use accounts without or with an empty password, the guest's group policy must be changed. To do so, open the group policy editor on the command line by typing `gpedit.msc`, open the key *Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options* and change the value of *Accounts: Limit local account use of blank passwords to console logon only* to *Disabled*.

- **Compacting virtual disk images is limited to VDI files.** The `VBoxManage modifyhd --compact` command is currently only implemented for VDI files. At the moment the only way to optimize the size of a virtual disk images in other formats (VMDK, VHD) is to clone the image and then use the cloned image in the VM configuration.

- **OVF import/export:**
  - OVF localization (multiple languages in one OVF file) is not yet supported.
  - Some OVF sections like StartupSection, DeploymentOptionSection and InstallSection are ignored.
  - OVF environment documents, including their property sections and appliance configuration with ISO images, are not yet supported.
  - Remote files via HTTP or other mechanisms are not yet supported.

- Neither **scale mode** nor **seamless mode** work correctly with guests using OpenGL 3D features (such as with compiz-enabled window managers).

- The RDP server in the VirtualBox extension pack supports only audio streams in format 22.05kHz stereo 16 bit. If the RDP client requests any other audio format there will be no audio.

- Preserving the aspect ratio in scale mode works only on Windows hosts and on Mac OS X hosts.

- On **Mac OS X hosts,** the following features are not yet implemented:
  - Numlock emulation
  - CPU frequency metric
  - Memory ballooning

- **Mac OS X guests:**
  - Mac OS X guests can only run on a certain host hardware. For details about license and host hardware limitations, please see chapter 3.1.1, *Mac OS X guests*, page 46 and check the Apple software license conditions.
  - VirtualBox does not provide Guest Additions for Mac OS X at this time.
  - The graphics resolution currently defaults to 1024x768 as Mac OS X falls back to the built-in EFI display support. See chapter 3.13.1, *Video modes in EFI*, page 59 for more information on how to change EFI video modes.
  - Mac OS X guests only work with one CPU assigned to the VM. Support for SMP will be provided in a future release.

  - Depending on your system and version of Mac OS X, you might experience guest hangs after some time. This can be fixed by turning off energy saving (set timeout to "Never") in the system preferences.
  - By default, the VirtualBox EFI enables debug output of the Mac OS X kernel to help you diagnose boot problems. Note that there is a lot of output and not all errors are fatal (they would also show on your physical Mac). You can turn off these messages by issuing this command:

    ```
    VBoxManage setextradata "VM name" "VBoxInternal2/EfiBootArgs" "  "
    ```

    To revert to the previous behavior, use:

    ```
    VBoxManage setextradata "VM name" "VBoxInternal2/EfiBootArgs" ""
    ```
  - It is currently not possible to start a Mac OS X guest in safe mode by specifying "-x" option in "VBoxInternal2/EfiBootArgs" extradata.

- **Solaris hosts:**
  - There is no support for USB devices connected to Solaris 10 hosts.
  - USB support on Solaris hosts requires Solaris 11 version snv_124 or higher. Webcams and other isochronous devices are known to have poor performance.
  - Host Webcam passthrough is restricted to 640x480 frames at 20 frames per second due to limitations in the Solaris V4L2 API. This may be addressed in a future Solaris release.
  - No ACPI information (battery status, power source) is reported to the guest.
  - No support for using wireless adapters with bridged networking.
  - Crossbow-based bridged networking on Solaris 11 hosts does not work directly with aggregate links. However, you can manually create a VNIC (using `dladm`) over the aggregate link and use that with a VM. This limitation does not exist in Solaris 11u1 build 17 and newer.

- **Guest Additions of version 4.1, 4.1.2 and 4.1.4 for Windows** Thus VirtualBox WDDM Video driver may be installed and kept in guest system if Guest additions uninstallation is performed. This is caused by a bug in Guest Additions uninstaller.

  > **Note:** This does **not** apply to Guest Additions update, i.e. installing a one version of Guest Additions on top of another works correctly.

  To solve this problem, one should uninstall the VirtualBox WDDM Video driver manually. To do that open Device Manager, and check whether the Display Adapter is named "VirtualBox Graphics Adapter ..". If no - there is nothing to be done. If yes - right-click the VirtualBox Graphics Adapter in Device Manager, select "Uninstall", check "Delete the driver software for this device" and click "OK". Once uninstallation is done - in Device Manager go to menu "Action" and select "Scan for hardware changes" to make the propper (Windows default) driver be picked up for the Graphics adapter.

- Neither *virtio* nor *Intel PRO/1000* drivers for **Windows XP guests** support segmentation offloading. Therefore Windows XP guests have slower transmission rates comparing to other guest types. Refer to MS Knowledge base article 842264 for additional information.

- **Guest Additions for OS/2.** Shared folders are not yet supported with OS/2 guests. In addition, seamless windows and automatic guest resizing will probably never be implemented due to inherent limitations of the OS/2 graphics system.

15 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 and second number represent the major version and the 3rd number the minor version. Minor version numbers of official releases are always even. An odd minor version number represents an internal development or test build. In addition, each build contains a revision number.

15.1 Version 5.1.22 (2017-04-28)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed *VERR_IEM_INSTR_NOT_IMPLEMENTED* Guru Meditation under certain conditions (5.1 regression; mostly Mac OS X hosts; bugs #15693, #15801, #16274, #16569, #16663)
- VMM: fixed software virtualization on Solaris hosts (5.1.20 regression)
- Storage: fixed a potential hang under rare circumstances (bug #16677)
- Storage: fixed a potential crash under rare circumstances (asynchronous I/O disabled or during maintenance file operations like merging snapshots)
- Linux hosts: make the ALSA backend work again as well as loading the GL libraries on certain hosts (5.1.20 regression; bugs #16667, #16693)
- Linux Additions: fixed mount.vboxsf symlink problem (5.1.20 regression; bug #16670)

15.2 Version 5.1.20 (2017-04-18)

This is a maintenance release. The following items were fixed and/or added:

- GUI: don't check if the Extension Pack is up-to-date if the user is about to install a new Extension Pack (bug #16317)
- GUI: fixed a possible crash when switching a multi-monitor VM into full-screen or seamless mode
- GUI: fixed non-literal shortcuts if the keyboard is not captured (5.1.10 regression; Windows hosts only)
- GUI: several mini-toolbar fixes in full-screen / seamless mode (X11 hosts only)
- GUI: don't crash on restoring defaults in the *appliance import* dialog
- Windows Additions: another fix for automatic logins for Windows Vista and newer (bug #15904)
- ICH9: fix for Windows guests with a huge amount (>64G) of guest memory
- BIOS: fixed El Torito hard disk emulation geometry calculation (thanks Dwight Engen)

15.3 Version 5.1.18 (2017-03-15)

This is a maintenance release. The following items were fixed and/or added:

- Shared Folders: fixed case insensitive filename access (5.1.16 regression; Windows guests only; bug #16549)
- Shared Folders: fixed access to long pathes (5.1.16 regression; Windows guests only; bugs #14651, #16564)
- API: fixed snapshot handling of medium attachments and PCI device attachments (bug #16545)
- API: make 32-bit Windows guests boot again with software virtualization if the ICH9 chipset is used (5.1.16 regression)
- VBoxBugReport: fixed VM log collection issue
- Linux hosts: fixed autostart service script (bug #14955)
- Windows Additions: fixed automatic logins for Windows Vista and newer (5.1.4 regression; bug #15904)

15.4 Version 5.1.16 (2017-03-08)

This is a maintenance release. The following items were fixed and/or added:

- VMM: don't access the *MSR_IA32_SMM_MONITOR_CTL* MSR if *dual-monitor treatment* is not available (KVM workaround, bug #14965)
- VMM: another fix for handling certain MSRs on ancient CPUs without VT-x support for MSR bitmaps
- VMM: fixed *VERR_SSM_LOAD_CPUID_MISMATCH* errors when restoring a saved state with SMP guests on hosts without the *CPUID/HTT* bit set (bug #16428)
- VMM: fixed a bug in call gate emulation
- VMM: *FWAIT* instruction fix
- VMM: fixed a sporadic guest hang under certain conditions
- GUI: hide the mini-toolbar from the taskbar and the pager on certain X11 hosts
- GUI: better error handling on the global settings / network / host-only / DHCP server settings
- GUI: fixes for full-screen with multiple screens
- Host-only Network: fixed host-only adapter creation issue preventing VirtualBox installation on Windows 10 hosts (bug #16379)
- NAT network: fixed two potential crashes in the DHCP server
- ICH9: fixed incorrect initialization of the primary bus for PCI bridges (5.1.14 regression)
- Storage: LsiLogic fix for Windows 10
- USB: fixed not being able to attach certain USB devices having invalid characters in the device strings (5.0.18 regression; bug #15956)

- USB: several fixes for the USB/IP support (bug #16462)
- VBoxSVC: fixed another crash during shutdown under rare circumstances
- VBoxSVC: fixed a stack overflow on (Windows debug builds only; bug #16409)
- OVF: when importing an appliance handle more than 10 network adapters if the OVA was created by VirtualBox (bug #16401)
- OVF: fixes for exporting and importing appliances with many disks (bug #16402)
- VBoxManage: fixed regression with *modifyhd –resize* (bug #16311)
- rdesktop-vrdp: source code tarball fixes
- Linux Installers: do not rebuild kernel modules unnecessarily (bug #16408)
- Linux hosts: added an action for opening the VM manager window to the .desktop file
- Linux hosts / guests: Linux 4.11 compile fixes (bug #16506)
- Linux Additions: added *vboxsf* FS modules alias (bug #16404)
- Linux Additions: fix for the shared folders kernel module to compile on Linux 4.10
- Linux Additions: properly install the Linux kernel module override rule on distributions without /etc/depmod.d
- Windows Additions: fixed a crash with recent Windows 10 builds if 3D is disabled (bug #15973)

15.5 Version 5.1.14 (2017-01-16)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed emulation of certain instructions for 64-bit guests on 32-bit hosts
- VMM: properly handle certain MSRs for 64-bit guests on ancient CPUs without VT-x support for MSR bitmaps (bug #13886)
- GUI: fixed a crash with multimonitor setups under certain conditions
- GUI: allow cloning of snapshots when the VM is running
- NVMe: fixed compatibility with the Storage Performance Development Kit (SPDK, bug #16368)
- VBoxSVC: fixed a crash under rare circumstances
- VBoxManage: added a sanity check to *modifymedium –resize* to prevent users from resizing their hard disk from 1GB to 1PB (bug #16311)
- Windows hosts: another fix for recent Windows 10 hosts
- Linux hosts: Linux 4.10 fixes
- Linux Additions: fixed protocol error during certain operations on shared folders (bug #8463)

15.6 Version 5.1.12 (2016-12-20)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed *VERR_IEM_ASPECT_NOT_IMPLEMENTED* Guru Meditations with certain Linux guests if KVM paravirtualization is enabled (5.1 regression; bugs #15613 and #16251)
- VMM: fixed *VERR_VMX_UNABLE_TO_START_VM* Guru Meditations under rare conditions
- GUI: prevent a crash under certain conditions if the VM is terminated very early
- GUI: fixed certain keyboard capture issues (5.1.10 regression; Mac OS X hosts only; bug #16150)
- GUI: fixed dragging guest windows in seamless mode with the keyboard captured (X11 hosts only; bug #15837)
- GUI: fixed a problem where the *new version detected* dialog was covered by the *appliance import* dialog (Mac OS X hosts only; bug #16238)
- Storage: fixed NVMe reset processing when doing *rmmod nvme; modprobe nvme* in a Linux guest (bug #16080)
- Storage: fixed creating a snapshot when the VM is running and an NVMe controller is present
- Storage: fixed a problem with the LsiLogic SCSI controller where requests could be lost with SMP guests
- E1000: fixed "cable disconnected" issue (Mac OS X guests only; 5.1.10 regression; bug #16260)
- E1000: fixed "TX unit hang" issue (Linux guests only; 5.1.10 regression; bug #16221)
- Parallel ports: fixed port enumeration on Windows host (bugs #15872 and #16127)
- API: don't crash when sanitizing certain VM names (bug #16299)
- Linux hosts: automatically disable asynchronous I/O on Linux 2.6.18 kernels as high I/O load may trigger kernel oopses on these kernels if this feature is enabled
- Linux hosts / guests: Linux 2.6.28 compile fix (bug #16267)
- Linux hosts: compile Linux 4.9 compile fix (bug #16286)
- Linux Additions: warn the user about a known bug with older Linux guests (e.g. Debian 7) requiring manual work to get 3D working (bug #15319)
- Linux Additions: fix the graphics driver build with Linux 4.10 and later (bug #16298)
- Windows Additions: fixed a crash in the WDDM driver under certain conditions

15.7 Version 5.1.10 (2016-11-21)

This is a maintenance release. The following items were fixed and/or added:

- GUI: the USB filter settings dialog should allow to specify the USB revision in hexadecimal format (bug #15400)

- GUI: fixed crash on certain hosts when pressing certain key combinations (Windows hosts only; bug #15719)
- GUI: fixed issue with updating the available-geometry on host-screen work-area resize
- GUI: don't crash / hang on certain environments if accessibility support is enabled
- GUI: fixed various issues in Unscaled HiDPI Output mode (bug #15707)
- GUI: extend the VM Input menu with *Print Screen*-related actions
- GUI: improved handling of inserting the Guest Additions ISO image by trying all available optical drives rather than only the first one and by not asking the user if he wants to force unmounting (which doesn't work in most cases anyway)
- API: default to RTC using UTC for Solaris 11 guests
- Settings: be less restrictive when reading a VM configuration containing a host-only adapter without an interface name
- Storage: fixed resizing VDI images resulting in an unbootable image under certain circumstances (bug #15983)
- NAT: fixed several 5.1.8 regressions on Mac OS X and Windows hosts (bug #16084)
- Audio: fixed a few 5.1.x regressions by using the audio code from 5.0.x until the audio overhaul is completed
- VBoxManage: fixed documentation of the *storagectl* command (bug #15971)
- Build system: another fix for building VirtualBox on systems which default to Python 3
- Windows hosts: hardening fix for Windows 10 build 14971 (bug #16202)
- Windows Additions: properly start the VirtualBox guest services even if the guest user name contains special characters (bug #15982)
- Solaris Additions: fixed preemptible mouse notification callback being executed under a spinlock for Solaris guests
- Linux hosts / guests: Linux 4.9 fixes (bugs #16155 and #16064)
- Linux Additions: fixed Linux kernel module override rule (thanks Mark Furneaux)

15.8 Version 5.1.8 (2016-10-18)

This is a maintenance release. The following items were fixed and/or added:

- GUI: fixed keyboard shortcut handling regressions (Mac OS X hosts only; bugs #15937 and #15938)
- GUI: fixed keyboard handling regression for separate UI (Windows hosts only; bugs #15928)
- NAT: don't exceed the maximum number of "search" suffixes. Patch from bug #15948.
- NAT: fixed parsing of port-forwarding rules with a name which contains a slash (bug #16002)

- NAT Network: when the host has only loopback nameserver that cannot be mapped to the guests (e.g. dnsmasq running on 127.0.1.1), make DHCP supply NAT Network DNS proxy as nameserver.
- Bridged Network: prevent flooding syslog with packet allocation error messages (bug #15569)
- Audio: now using Audio Queues on Mac OS X hosts
- Audio: fixed recording with the PulseAudio backend (5.1 regression)
- Audio: various bugfixes
- Snapshots: fixed regression in 5.1.4 for deleting snapshots with several disks (bug #15831)
- Snapshots: crash fix and better error reporting when snapshot deletion failed
- Storage: some fixes for the NVMe emulation with Windows guests
- API: fixed initialization of SAS controllers (bug #15972)
- Build system: make it possible to build VBox on systems which default to Python 3
- Windows hosts: detect certain cases of *REGDB_E_CLASSNOTREG* errors and print a helpful error message
- Windows hosts: adapted to changes in Windows 10 build 14901 (bug #15944)
- Windows hosts: better support for processor groups on Windows 7 and later which is required on certain hosts with many CPUs
- Windows installer / Additions: added option to prevent creating of start menu items (bug #15922)
- Windows Additions / VGA: if the guest's power management turns a virtual screen off, blank the corresponding VM window rather than hide the window
- Windows Additions: fixed a generic bug which could lead to freezing shared folders (bug #15662)
- Linux hosts / guests: fix for kernels with *CONFIG_CPUMASK_OFFSTACK* set (bug #16020)
- Linux Additions: don't require all virtual consoles be in text mode. This should fix cases when the guest is booted with a graphical boot screen (bug #15683)
- Linux Additions: added depmod overrides for the vboxguest and vboxsf kernel modules to fix conflicts with modules shipped by certain Linux distributions
- X11 Additions: disable 3D on the guest if the host does not provide enough capabilities (bug #15860)

15.9 Version 5.1.6 (2016-09-12)

This is a maintenance release. The following items were fixed and/or added:

- GUI: fixed issue with opening '.vbox' files and it's aliases
- GUI: keyboard grabbing fixes (bugs #15771 and #15745)
- GUI: fix for passing through Ctrl + mouse-click (Mac OS X hosts only; bug #15714)

- GUI: fixed automatic deletion of extension pack files (bugs #11352 and #14742)
- USB: fixed showing unknown device instead of the manufacturer or product description under certain circumstances (5.1.0 regression; bug #15764)
- XHCI: another fix for a hanging guest under certain conditions as result of the fix for bug #15747, this time for Windows 7 guests
- Serial: fixed high CPU usage with certain USB to serial converters on Linux hosts (bug #7796)
- Storage: fixed attaching stream optimized VMDK images (bug #14764)
- Storage: reject image variants which are unsupported by the backend (bug #7227)
- Storage: fixed loading saved states created with VirtualBox 5.0.10 and older when using a SCSI controller (bug #15865)
- Storage: fixed broken NVMe emulation if the host I/O cache setting is enabled
- Storage: fixed using multiple NVMe controllers if ICH9 is used
- NVMe: fixed a crash during reset which could happen under certain circumstances
- Audio: fixed microphone input (5.1.2 regression; bugs #14386 and #15802)
- Audio: fixed crashes under certain conditions (5.1.0 regression; bug #15887 and others)
- Audio: fixed recording with the ALSA backend (5.1 regression)
- Audio: fixed stream access mode with OSS backend (5.1 regression, thanks to Jung-uk Kim)
- E1000: do also return masked bits when reading the ICR register, this fixes booting from iPXE (5.1.2 regression; bug #15846)
- BIOS: fixed 4bpp scanline calculation (bug #15787)
- API: relax the check for the version attribute in OVF/OVA appliances (bug #15856)
- Windows hosts: fixed crashes when terminating the VM selector or other VBox COM clients (bug #15726 and others)
- Linux Installer: fixed path to the documentation in .rpm packages (5.1.0 regression)
- Linux Installer: fixed the vboxdrv.sh script to prevent an SELinux complaint (bug #15816)
- Linux hosts: don't use 32-bit legacy capabilities
- Linux Additions: Linux 4.8 fix for the kernel display driver (bugs #15890 and #15896)
- Linux Additions: don't load the kernel modules provided by the Linux distribution but load the kernel modules from the official Guest Additions package instead (bug #15324)
- Linux Additions: fix dynamic resizing problems in recent Linux guests (bug #15875)
- User Manual: fixed error in the VBoxManage chapter for the *getextradata enumerate* example (bug #15862)

15.10 Version 5.1.4 (2016-08-16)

This is a maintenance release. The following items were fixed and/or added:

- GUI: show actual VM uptime in the session information window
- Audio: re-enabled speakers for Mac OS X guests (5.1.0 regression; bug #15611)
- Audio: fixed crashes under certain conditions
- USB: fixed a hang under certain conditions
- USB: fixed a hanging guest under certain conditions (bug #15747)
- PIIX4: implemented dummy SMBus controller to prevent annoying Linux kernel warnings about uninitialized SMBus base address (bug #9517)
- NVMe: several fixes to improve stability, fixed a crash while saving a VM state
- VMDK: fixed an issue creating fixed size images with certain sizes and the Split2G option enabled (bug #15748)
- VHDX: fixed cloning images with VBoxManage clonehd (bug #14288)
- Storage: fixed broken bandwidth limitation when the limit is very low (bug #14982)
- EFI: fixed sending debug messages in the EFI firmware if a serial port is enabled (bug #12161)
- OVF: when importing appliances, make sure that the version of the embedded VirtualBox specific settings is processed, to get the default settings handling right
- VBoxManage: Don't try to set the medium type if there is no change (bug #13850)
- Linux installer: fixed some scripting issues (bugs #15701 and #15702)
- Linux installer: fixed a path issue on certain Linux distributions (bug #15717)
- Windows hosts: fixed corrupted mouse pointers with some Linux and Solaris guests (bug #15665)
- Linux Additions: made the video driver work on 32-bit guests with large video memory sizes (bug #15621)
- Linux Additions: made the video driver work on kernel 4.7 and later (bug #15769)
- Linux Additions: converted a failure message to an informational one when drivers could not be stopped during upgrade (bug #15692)
- Linux Additions: made the video driver work around an X server bug which cause screen refresh to stop (bug #15511)
- Windows Additions: auto-resizing fixes for Windows 10 guests (bug #15257)
- Windows Additions: fixed VBoxTray problems with Windows 2000 (bug #15661)

15.11 Version 5.1.2 (2016-07-21)

This is a maintenance release. The following items were fixed and/or added:

- VMM: several fixes
- GUI: fixed screenshot if the VM is started in separate mode
- GUI: fixed issue with double-click/opening registered file types (.vbox and .vbox-extpack) on Mac OS X (bug #15648)
- GUI: fixed a bug which made it impossible close some error/warning messages using the close button
- GUI: weakened the control over the VM window behavior allowing to resize it across multiple screens and to use in Snap Assist techniques
- GUI: jump to the second tab of the session information window (5.1.0 regression)
- GUI: fix for Alt-Tab
- Storage: fixed creating fixed size VHD images (bug #15601)
- Storage: fixed a hang during power off if the VM was suspended before and a NVMe controller is configured
- USB: fixed a crash under certain conditions
- Audio: make AC'97 volume control work again (5.1.0 regression; bug #15598)
- Audio: fixed rare VM hangs when using AC'97 emulation
- Audio: SB16 fixes
- EFI: fixed access to devices attached to SATA port 2 and higher (bug #15607)
- OVA: fix for checking certain signatures
- OVA: fixed MAC address generation for appliances created by VirtualBox (5.1.0 regression; bug #15623)
- API: fixed audio settings handling for older config files (bug #15626)
- API: fixed truncation of USB product/vendor IDs on Linux hosts (5.1.0 regression; bug #15644)
- API: fixed VRDP with authentication (bug #15653)
- API: don't crash if there is no graphics controller configured (bug #15628)
- Linux hosts: fixed EL5 builds (bug #15634)
- non-Windows hosts: fixed a crash during shutdown under rare circumstances (bug #15568)
- Linux Additions: fixed SELinux issue which prevented certain Linux guests to work in 3D mode (bug #15574)
- User Manual: updates

15.12 Version 5.1.0 (2016-07-12)

This is a major update. The following major new features were added:

- VMM: new APIC and I/O APIC implementations that result in significantly improved performance in certain situations (for example with networking, bug #15295)
- VMM: added support for Hyper-V paravirtualized debugging of Windows guests
- VMM: emulate even more MMIO and shadow pagetable exits without going back to user mode
- GUI: overall migration to Qt5 (bug #11775)
- GUI: passive API event listener improving the VM GUI performance and response time
- Audio: added HDA (High Definition Audio) support for newer Linux guests
- Audio: added on-demand timers which should improve the overall performance and reduce the CPU consumption
- Audio: more fine-grained volume control for the AC'97 emulation, which now also takes the master volume control into account
- better support for Python 3

In addition, the following items were fixed and/or added:

- VMM: activate the x2APIC by default for Linux guests
- VMM: many more fixes
- GUI: the *Detach UI* action is now a part of the VM Machine menu
- GUI: reworked session information window
- GUI: the new VM wizard now allows to choose the VM location for the VM which is being created
- GUI: fixed location of touch events in unscaled HiDPI mode (bugs #14366, #14367)
- GUI: various bugfixes and internal cleanup
- Storage: use *fallocate()* or similar if present to speed up creating fixed-sized disk images (bug #5967)
- Storage: implemented NVMHCI-compatible storage controller
- Audio: various bugfixes and infrastructure improvements
- E1000: implemented interrupt throttling for performance improvements
- EFI: many improvements, for example allow to boot from USB
- OVF: improved handling of signed manifests
- API: Windows XP guests and Windows 2003 server guests now default to the E1000 T Server network card
- VBoxSVC: fixed several memory leaks when handling .dmg images
- Installers: ship VBoxBugReport

- Windows hosts: reworked COM proxy (e.g. bug #8298)
- Linux installers: no longer rely on DKMS for module rebuilding
- Linux Additions: fix a problem when updating the initramfs after the guest modules are compiled (bug #15579)
- Linux Additions: try to fix a problem where the OpenGL libraries where loaded by the X server when we only support X clients (bug #15574)

15.13 Version 5.0.24 (2016-06-28)

This is a maintenance release. The following items were fixed and/or added:

- VMM: reverted to the old I/O-APIC code for now to fix certain regressions with 5.0.22 (bug #15529). This means that the networking performance with certain guests will drop to the 5.0.20 level (bug #15295). One workaround is to disable GRO for Linux guests.
- Main: when taking a screenshot, don't save garbage for blanked screens
- NAT: correctly parse resolv.conf file with multiple separators (5.0.22 regression)
- Storage: fixed a possible corruption of stream optimized VMDK images from VMware when opened in read/write mode for the first time
- ACPI: notify the guest when the battery / AC state changes instead of relying on guest polling
- Linux hosts: fixed *VERR_VMM_SET_JMP_ABORTED_RESUME* Guru Meditations on hosts with Linux 4.6 or later (bug #15439)
- Solaris hosts: make the GUI work on Solaris 10 again (bug #15549)

15.14 Version 5.0.22 (2016-06-16)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixes for certain Intel Atom hosts (bug #14915)
- VMM: properly restore the complete FPU state for 32-bit guests on 64-bit hosts on Intel Sandy Bridge and Ivy Bridge CPUs
- VMM: new I/O-APIC implementation fixing several bugs and improving the performance under certain conditions (bug #15295 and others)
- VMM: fixed a potential Linux guest panic on AMD hosts
- VMM: fixed a potential hang with 32-bit EFI guests on Intel CPUs (VT-x without unrestricted guest execution)
- GUI: don't allow to start subsequent separate VM instances
- GUI: raised upper limit for video capture screen resolution (bug #15432)
- GUI: warn if the VM has less than 128MB VRAM configured and 3D enabled
- Main: when monitoring DNS configuration changes on Windows hosts avoid false positives from competing DHCP renewals. This should fix NAT link flaps when host has multiple DHCP configured interfaces, in particular when the host uses OpnVPN.

- Main: properly display an error message if the VRDE server cannot be enabled at runtime, for example because another service is using the same port
- NAT: Initialize guest address guess for wildcard port-forwarding rules with default guest address (bug #15412)
- VGA: fix for a problem which made certain legacy guests crash under certain conditions (bug #14811)
- OVF: fixed import problems for some appliances using an AHCI controller created by 3rd party applications
- SDK: reduced memory usage in the webservice Java bindings
- Windows hosts: fixed performance regresson with SMP guests (5.0 regression)
- Windows hosts: fixes for the shared clipboard
- Windows hosts: Windows hardening fix
- Windows Additions: fixes to retain the guest display layout when resizing or disabling the guest monitors
- Linux hosts: EL 6.8 fix (bug #15411)
- Linux hosts: Linux 4.7 fix (bug #15459)
- Linux Additions: Linux 4.7 fixes (bug #15444)
- Linux Additions: fix for certain 32-bit guests (5.0.18 regression; bug #15320)
- Linux Additions: fixed mouse pointer offset (5.0.18 regression; bug #15324)
- Linux Additions: made old X.Org releases work again with kernels 3.11 and later (5.0.18 regression; bug #15319)
- Linux Additions: fixed X.Org crash after hard guest reset (5.0.18 regression; bug #15354)
- Linux Additions: don't stop the X11 setup if loading the shared folders module fails (5.0.18 regression)
- Linux Additions: don't complain if the Drag and Drop service is not available on the host
- Solaris Additions: added support for X.org 1.18

15.15 Version 5.0.20 (2016-04-28)

This is a maintenance release. The following items were fixed and/or added:

- Storage: fixed a regression causing write requests from the BIOS to cause a Guru Meditation with the LsiLogic SCSI controller (5.0.18 regression; bug #15317)
- Storage: several emulation fixes in the BusLogic SCSI controller emulation
- NAT Network: support TCP in DNS proxy (same problem as in bug #14736 for NAT)
- NAT: rework handling of port-forwarding rules (bug #13570)
- NAT: rewrite host resolver to handle more query types and make it asynchronous so that a stalled lookup doesn't block all NAT traffic

- Snapshots: don't crash when restoring a snapshot which has more network adapters than the current state (ie when the snapshot uses ICH9 and the current state uses PIIX3)
- Guest Control: various bugfixes for the *copyfrom* and *copyto* commands / API (bug #14336)
- VBoxManage: list processor features on *list hostinfo* (bug #15334)
- Linux hosts: fix for Linux 4.5 if *CONFIG_NET_CLS_ACT* is enabled (bug #15327)
- Windows hosts: another fix for recent Windows 10 insider builds (bug #15337)
- Windows hosts: make it work on Windows XP again (5.0.18 regression)
- Windows Additions: fixed performance issues with PowerPoint 2010 and the WDDM graphics drivers if Aero is disabled

15.16 Version 5.0.18 (2016-04-18)

This is a maintenance release. The following items were fixed and/or added:

- GUI: position off-screen windows to be fully visible again on relaunch in consistence with default-behavior (bug #15226)
- GUI: fixed the *View* menu / *Full-screen Mode* behavior on Mac OS X El Capitan
- GUI: fixed a test which allowed to encrypt a hard disk with an empty password
- GUI: fixed a crash under certain conditions during VM shutdown
- GUI: fixed the size of the VM list scrollbar in the VM selector when entering a group
- PC speaker passthrough: fixes (Linux hosts only; bug #627)
- Drag and drop: several fixes
- SATA: fixed hotplug flag handling when EFI is used
- Storage: fixed handling of encrypted disk images with SCSI controllers (bug #14812)
- Storage: fixed possible crash with Solaris 7 if the BusLogic SCSI controller is used
- USB: properly purge non-ASCII characters from USB strings (bugs #8801, #15222)
- NAT Network: fixed 100% CPU load in VBoxNetNAT on Mac OS X under certain circumstances (bug #15223)
- ACPI: fixed ACPI tables to make the display color management settings available again for older Windows versions (4.3.22 regression)
- Guest Control: fixed *VBoxManage copyfrom* command (bug #14336)
- Snapshots: fixed several problems when removing older snapshots (bug #15206)
- VBoxManage: fixed *–verbose* output of the *guestcontrol* command
- Windows hosts: hardening fixes required for recent Windows 10 insider builds (bugs #15245, #15296)
- Windows hosts: fixed support of jumbo frames in with bridged networking (5.0.16 regression; bug #15209)

- Windows hosts: don't prevent receiving multicast traffic if host-only adapters are installed (bug #8698)
- Linux hosts: added support for the new naming scheme of NVME disks when creating raw disks
- Solaris hosts / guests: properly sign the kernel modules (bug #12608)
- Linux hosts / guests: Linux 4.5 fixes (bug #15251)
- Linux hosts / guests: Linux 4.6 fixes (bug #15298)
- Linux Additions: added a kernel graphics driver to support graphics when X.Org does not have root rights (bug #14732)
- Linux/Solaris Additions: fixed several issues causing Linux/Solatis guests using software rendering when 3D acceleration is available
- Windows Additions: fixed a hang with PowerPoint 2010 and the WDDM drivers if Aero is disabled

15.17 Version 5.0.16 (2016-03-04)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed a problem which could lead to a wrong guest behavior on AMD CPUs (bugs #14831 and #15186)
- GUI: don't try to synchronize the HID LEDs if the VM window is not active or if it's minimized (Windows / Mac OS X hosts only; bug #14302)
- GUI: prevent a crash during startup under rare conditions
- GUI: sub-menu option to disable the guest-OS type overlay in the application dock icon on Mac OS X
- GUI: position off-screen windows to be fully visible again on relaunch
- GUI: hide the VT-x/AMD-V checkbox if raw-mode is not supported (usually Mac OS X hosts; bug #15178)
- PC speaker passthrough: new experimental feature, available on Linux host only
- Audio: several fixes for Mac OS X hosts + guests
- Audio: properly handle default audio device changes (Windows hosts)
- USB: serveral fixes for the xHCI controller (e.g. for webcam passthrough)
- BIOS: fixed int15/AH=83/AL=00 function (4.2.0 regression)
- iPXE: enable the HTTP download protocol on non-Linux hosts (bug #13628)
- Shared folders: fixed a failure to load the saved state under certain circumstances (bug #6314)
- Guest Control: added support for (cached) Active Directory authentication in case the domain controller is not reachable (anymore)
- Serial ports: raised the number of serial ports per VM from 2 to 4 (bug #9109)

- Serial ports: fix for the TCP/IP backend (Windows hosts only; bug #15188)
- SDK: make the Python webservice API binding work again (5.0 regression)
- Seamless mode: fixed a crash under certain circumstances (bug #15106)
- Linux hosts: fixed the /sbin/rcvboxdrv script as well as the missing shebang in two scripts (bugs #15055 and #15057)
- Linux hosts: properly uninstall Python files installed by the .run installer
- Windows hosts: hardening fix required for recent Windows insider builds (bug #14052)
- Windows hosts: fixed Python installation path (bug #13131)
- Windows hosts: support MTU larger than 2 KB with bridged networking (bug #15140)
- Windows hosts / guests: properly sign binaries using a sha-256 certificate (bug #15054)
- Windows Additions: fixed guest property enumeration of logged-in users
- Windows Additions: fixed sporadical failure of the graphics driver in Windows 10 guests (bug #14409)
- Windows Additions: under rare circumstances no mouse movement events were delivered to the guest

15.18 Version 5.0.14 (2016-01-19)

This is a maintenance release. The following items were fixed and/or added:

- GUI: properly limit the number of VCPUs to the number of physical cores on Mac OS X (bug #15018)
- Audio: fixed a bug which prevented loading a saved state of a saved guests with HDA emulation (5.0.12 regression; bug #14981)
- Audio: don't crash if the backend is unable to initialize (bug #14960)
- Audio: fixed audio capture on Mac OS X (bug #14386)
- Storage: fixed a possible crash when attaching the same ISO image multiple times to the same VM (bug #14951)
- BIOS: properly report if two floppy drives are attached
- USB: fixed a problem with filters which would not capture the device under certain circumstances (5.0.10 regression; bug #15042)
- ExtPack: black-list Extension Packs older than 4.3.30 due to incompatible changes not being properly handled in the past
- Windows hosts: fixed a regression which caused robocopy to fail (bug #14958)
- Linux hosts: properly create the */sbin/rcvboxdrv* symbolic link (5.0.12 regression; bug #14989)
- Mac OS X hosts: several fixes for USB on El Capitan (bug #14677)
- Linux Additions: fixes for Linux 4.5 (bug #15032)

15.19 Version 5.0.12 (2015-12-18)

This is a maintenance release. The following items were fixed and/or added:

- GUI: fixed wrong scrolling behaviour in the VM selector window when a VM item is dragged out of the chooser-pane area
- GUI: fixed the validation of IPv6 port-forwarding rules
- GUI: suppress the first-run wizard if a CD/DVD medium is inserted using the selector UI
- GUI: fixed the Ctrl+Break key sequence scan codes (bug #14927)
- GUI: improved handling of text selection mouse pointer (bug #750)
- Host services: fixed a crash during VM shutdown under rare conditions (5.0.6 regression; bug #14841)
- Shared folders: fixed a sharing violation if a file is opened to check the attributes (Windows hosts only; bug #14450)
- Webcam: passthrough fix for certain devices (Mac OS X hosts only)
- XHCI: fixed broken emulation if software virtualization is used
- XHCI: several fixes
- 3D: fixed state handling under certain conditions (bug #13487)
- Audio: several fixes
- BIOS: added LBA64 support for being able to boot from huge hard disks(bug #7415)
- EFI: fix for Windows 10 guests
- ExtPack: before installing an Extension Pack check if there are VMs running to prevent file system locking issues
- rdesktop-vrdp: source code tarball fixes
- Windows hosts: fixed hang when using VBoxAuthSimple library for VRDP external authentication (bug #14931)
- Windows hosts: fixed a regression which prevented it to attach to a physical network adapter having TCP/IP disabled (bug #14578)
- Windows hosts: fixed a regression which caused multi-port adapters to be shown as a single adapter (bugs #14558, #14622)
- Windows hosts: fixed a regression which caused created host-only adapters to not appear in the list (bug #14437)
- Windows hosts: fixed host-only adapter creation issues related to Windows 10 (bugs #14040, #14545)
- Linux hosts: .desktop file compatibility issue (bug #14808)
- Linux hosts / guests: fixes for RHEL 7.2 (bug #14866)
- Linux hosts: the command for recompiling the host kernel modules was changed again, to */sbin/rcvboxdrv setup* (bug #14723)

- Linux hosts: some fixes for PCI passthrough (still highly experimental)
- Linux/Mac OS X hosts: fixed a VM hang during startup under certain circumstances (bug #14933)
- Solaris hosts: added Python 2.7 bindings
- Mac OS X hosts: fixed a possible crash when the default input or output audio device changes
- Mac OS X hosts: fixed a panic under certain conditions
- Linux Additions: prevent the compiler from doing dead-code elemination on vital code in guest / host communication (bug #14497)
- Linux Additions: when mounting a shared folder, explicitly pass the share name so that /proc/mounts contains this name instead of 'none'
- Linux Additions: workaround for a systemd problem in conjunction with SELinux which prevented to properly enable the 'vboxadd' service during while upgrading the Additions

15.20 Version 5.0.10 (2015-11-10)

This is a maintenance release. The following items were fixed and/or added:

- VMM: improved support for certain Intel Atom CPUs (bug #14773)
- VMM: system register emulation fix (5.0 regression; bug #14515)
- GUI: fixed immediate screenshot issue (bug #14108)
- GUI: fixed another 3D overlay window reparenting issue when the VM is switched to full-screen mode on X11 hosts
- GUI: fixed help index (bug #14722)
- GUI: fixed state synchronization issue in the VM manager window when VM was paused from its runtime window
- Audio: fixed suspending/resuming audio streams on VM pause/unpause (bug #14784)
- Audio: properly reset AC'97 audio streams, otherwise there is silence until a non-48 kHz stream is played
- Audio: fixed a small emulation quirk of the AD1980 codec of the HDA device to make recent linux guests work (bug #14653)
- USB: serveral fixes for the xHCI controller
- USB: fixed a crash under certain conditions on hosts with Linux kernels older than version 3.3
- USB: better identification of certain USB devices
- NAT: support TCP in DNS proxy (bug #14736)
- NAT Network: fixed sporadic crashes on Windows hosts (bug #13899)
- API: when creating differencing images (e.g. as part of a snapshot or cloning a VM) use the same disk image variant as the parent image if possible, which means that e.g. a diff image for a VMDK image split into 2 GB files will also be split (bug #14764)

- API: event queue handling fixes preventing loss of certain events at runtime (e.g. new webcam attached), particularly important on Mac OS X hosts
- Webcam: passthrough fix for certain devices (Windows hosts only)
- VBoxManage: don't crash on *snapshot restorecurrent / edit* if the VM has no snapshots
- VBoxManage: don't crash on *controlvm addencpassword* (bug #14729)
- Mac OS X hosts: use the correct kernel on certain hosts
- Windows hosts: fixed VRDP external authentication
- Windows hosts: allow to use a shared folder path with extended-length path prefix (5.0 regression; bug #14651)
- Windows hosts: fix a crash in the netfilter host driver under certain conditions (bug #14799)
- Windows host installer: documented and fixed public properties which can be used to control the installation to some extent
- Windows host installer: fixed not starting the actual installation when showing the version information or help dialogs
- X11 Additions: added basic support for X.Org Server 1.18 (3D requires additional fixes)

15.21 Version 5.0.8 (2015-10-20)

This is a maintenance release. The following items were fixed and/or added:

- GUI: Mac OS X: Restore green zoom button for VM windows (it was hidden in previous release to avoid native full-screen issues). For Yosemite and El Capitan this button should work accordingly to the Apple HIG: Full-screen by default, maximize if the user holds the Option key.
- Serial ports: fixed wrong IRQ number for the first serial port in the ACPI tables (5.0.6 regression; bug #14659)
- API: fixed a 5.0 regression in *VBoxManage setproperty defaultfrontend* (bug #14696)
- VBoxManage/vbox-img: conversion to RAW images could result in a disk image containing all zeroes
- Linux hosts: several fixes for systemd integration in .deb / .rpm packages (e.g. bug #14665). The command for recompiling the host kernel modules was changed to */sbin/vboxconfig*
- Linux hosts: make host-only interfaces report operstate UP only when they have VMs attached (bug #14526)
- Mac OS X hosts: fix bpf capture and accounting of traffic on bridged and host-only interfaces (bug #14553)
- Windows guests: fixed 3D rendering issues on high resolution displays
- Windows Additions: fixed problems with 3D acceleration on Windows hosts with Intel HD graphics (bug #14670)
- Linux Additions: fix service starting on Debian systems with systemd installed but not in use (bug #14683)

15.22 Version 5.0.6 (2015-10-02)

This is a maintenance release. The following items were fixed and/or added:

- GUI: the update check now uses the HTTP system proxy settings by default
- GUI: About dialog improvements. Copyable version text, do not close dialog on mouse-clicks and focus losing, explicit close button at the bottom of dialog and disabled close button fix on OS X. (bugs #9912, #12749)
- GUI: fixed bug when re-assigning shortcuts (bug #14565)
- GUI: fixed default focus button in message-box dialogs (bug #14486)
- GUI: fixed settings dialog which is opened if the network settings need to be changed at VM startup (5.0 regression; bug #14601)
- GUI: fixed crash during VM start if an early error message needs to be shown, for example Linux kernel modules not present (bug #14646)
- Bridged Networking: fixed handling of guest DHCP requests without UDP checksum when bridging to a wireless interface (bug #14615).
- Audio: latency fixes (Windows hosts only; bug #4088)
- Guest Control: correctly set *USERNAME* and *USERPROFILE* environment variables (Windows guests only)
- Guest Control: several fixes
- API: properly restore NAT port forwarding rules when reverting to a snapshot
- Parallel ports: Several fixes allowing to enable two parallel ports for a VM
- VBoxManage: fixed wrong output of *debugvm show* command
- VBoxManage: fixed hang when specifying logging groups with *debugvm log* starting with *h*, for example *hex*
- Windows hosts: renamed *VBoxStartup.log* to *VBoxHardening.log* and provide this log file in the GUI log viewer
- Windows hosts: fixed a small memory leak in the Windows host interface driver (VBoxNetAdp) which caused a BSOD if the driver verifier is enabled (bug #14562)
- Windows hosts: fixed a failure to start VMs on hosts where dsound.dll is not available (bug #14574)
- Windows hosts: another fix for *VERR_LDR_MISMATCH_NATIVE* errors (bug #14579)
- Windows hosts: fixed host-to-guest communication with bridged networking (bugs #14326, #14457)
- Windows hosts: fixed broken data receiving from the serial device with the named pipe backend if Kaspersky AV is installed
- Linux hosts: Linux 4.3 compile fixes
- Linux hosts: installer fix for certain systems (bug #14627)

- Linux hosts / guests: native systemd support for the host/guest installer scripts. The scripts for re-compiling the kernel modules are now located at /sbin/rcvboxdrv (host) and /sbin/rcvboxadd (Guest Additions)
- Mac OS X hosts: GUI-related fixes for El Capitan
- Mac OS X hosts: fixed a problem with capturing USB devices under El Capitan
- Mac OS X hosts: allow colon character on shared folders (bug #14554)
- Linux Additions: properly set the VBoxService process ID in the PID file (bug #14571)
- Linux Additions: Guest Control fixes (bug #14573)
- Windows Additions: fixed shutting down VBoxTray when running with older VirtualBox host versions
- Windows Additions: fixed video playback with VLC and Windows Media Player when the WDDM driver is used and 3D is not used
- Windows Additions: prevent a possible VLC crash when the WDDM driver is used and 3D is enabled by implementing YV12 surfaces

15.23 Version 5.0.4 (2015-09-08)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed an issue with Windows 10 guest kernel debugging over the network for Hyper-V paravirtualized VMs
- VMM: fixed a bug which prevented reading the saved state of the 'PATM' unit from VirtualBox 4.3.x (bug #14512)
- GUI: changed default OS type for Windows from Windows XP to Windows 7
- GUI: added another pre-defined guest screen resolution (bug #14384)
- GUI: fixed update check which was broken due to changing the location of the root certificates (bug #13096)
- GUI: fixed issues with synchronization of Caps lock / Num lock / Scroll lock on Windows hosts (bug #14302)
- GUI: don't crash during VM shutdown if 2D video acceleration and 3D support are enabled (Mac OS X hosts only)
- GUI: several seamless fixes for certain X11 window managers, also when used in multi-screen setups
- GUI: Log window size, position and cursor-position fixes
- Audio: fixed playing leftover/deprecated audio samples
- Audio: fixed playing audio after suspending the host (5.0 regression; Linux hosts using the ALSA backend)
- Audio: fixed playing short audio samples which were chopped off formerly
- Audio: fixed distortions on OS X when the sample rate of the guest stream and host device don't match

- Storage: fixed raw disk access and flat VMDK image access which would be always opened readonly (5.0.2 regression; bugs #14425, #14461)
- Storage: fixed initial encryption of VDI images after they were compacted (bug #14496)
- VGA: fix for certain graphics modes (bug #14516)
- NAT: don't freeze while the VM is paused if the network attachment mode is changed from/to NAT with activated port forwarding
- OVF: fixed duplicate USB controller entries in exported OVA/OVF (bug #14462)
- Shared Folders: fixed a path separator issue (bug #14434)
- Drag and drop: fixed crashes on OS X hosts when doing host-to-guest transfers
- VBoxManage: another attempt to not deny changing the network adapter type at VM runtime (5.0 regression; bug #14308)
- VBoxManage: fixed broken *guestcontrol <VM-Name> list* command (5.0 regression)
- VBoxManage: fixed broken Guest Control stdout/stderr output (5.0 regression)
- Mac OS X hosts: fixed remaining problems with activated SMAP (Broadwell and later; bug #14412)
- Mac OS X hosts: fixed broken 3D support (5.0.2 regression; bug #14476)
- Linux hosts: Linux 4.2 fix
- Linux hosts: don't crash on older Linux distributions if the DBus service isn't running (bug #14543)
- Windows hosts: fixed the *VERR_LDR_MISMATCH_NATIVE* error message (bug #14420)
- Windows hosts: fix for Windows 10 build 10525 and later (bug #14502)
- Windows hosts: fixed network adapter enumeration on Windows 10 (bug #14437)
- Windows hosts: prevent intermittent host network disconnects during VM start/shutdown with bridged networking (bug #14500)
- Windows Additions: fixed the call to the memory allocation function (bug #14415)
- Linux Additions: be more forgiving if the compilation of the vboxvideo module fails (bug #14547)
- X11 Additions: fixed a number of small issues with dynamic resizing and full-screen and seamless modes.

15.24 Version 5.0.2 (2015-08-13)

This is a maintenance release. The following items were fixed and/or added:

- VMM: added support for guest crash report MSRs with Hyper-V paravirtualization
- VMM: fixed an issue causing artificially high load averages on Linux hosts
- VMM: fixed a kernel panic with thread-context hooks caused by incompatible changes made to Linux 4.2 kernels

- VMM: fixed a saved state issue with VT-x/AMD-V disabled (5.0 regression; bug #14304)
- VMM: fixed *VERR_SUPDRV_TSC_DELTA_MEASUREMENT_FAILED* Guru Meditations on certain AMD CPUs (5.0 regression; bug #14370)
- VMM: fixed a crash while creating a guest core dumps via the VM debug facility (5.0 regression)
- VMM: This release has AVX2 passthrough disabled on every host and AVX passthrough disabled for 64-bit VMs on 32-bit hosts. This will be properly fixed in a future 5.0.x maintenance release (see e.g. bug #14262).
- GUI: fixed rare hang and crash on VM shutdown/poweroff
- GUI: X11: fixed few crashes caused by the Qt *alien widgets* feature
- GUI: X11: fixed various mini-toolbar geometry quirks like positioning, z-order, transparency issues on certain window managers (bug #14257)
- GUI: X11: fixed mini-toolbar minimize button issue under certain window managers (bug #14284)
- GUI: VM menu actions availability should now be properly updated on full-screen/seamless/scaled mode switches
- GUI: disk encryption password validation should be performed when user confirmed the password, not after each entered symbol
- GUI: do not change the VM/group selection in the VM Manager to the newly created VM if it was created by another client (e.g. VBoxManage)
- GUI: Mac OS X: do not treat 'almost maximized' VM windows as 'maximized', watch for the strict window geometry instead
- GUI: improve the quality in scaled mode under some circumstances (5.0 regression; bug #14303)
- VBoxManage: do not deny changing the network adapter type at VM runtime (5.0 regression; bug #14308)
- VRDP: allow Windows 10 RDP clients (bug #14216)
- Audio: fix a possible crash on VM process termination (5.0 regression)
- Storage: improved raw disk access on OS X by unmounting any accessed volume before first use and prevent any mount attempt by the host (bug #14219)
- 3D: basic support for saving/restoring display lists
- Drag and drop: fixed guest to host transfers on OS X hosts
- Drag and drop: fixed memory leak on Windows guests
- Shared Folders: fixed a problem with accessing CIFS shares (bug #14252)
- Shared Folders: improved path conversion between hosts and guests with different path separators (bug #14153)
- API: skip resetting of immutable media when the VM in saved state is started (bug #13957)
- API: fixed method for setting medium IDs which used zero (invalid) UUIDs instead random (valid) UUIDs if no UUIDs were passed (bug #14350)

- API: for Windows host fix detection of API client crashes which have a session open
- OVF: properly export all VBox features including the setting for paravirtualization (bug #14390)
- Mac OS X hosts: El Capitan USB fixes
- Windows hosts: fixed crash when opening Windows dialogs from the VM process on Windows 10 (bug #14351)
- Windows hosts: fixed host-only adapter creation issues on Windows 10 (bug #14040)
- Windows hosts: fixed audio on Windows 10 (bug #14432)
- Linux hosts: more fixes for activated SMAP on Linux 3.19 and newer (Broadwell and later; bug #13961)
- Linux hosts: check then name space before attaching to a host network interface (bug #13795)
- Linux Additions: Linux 4.2 fixes (bug #14227)
- Linux Additions: improved the performance of *stat()* to speed up certain file operations on shared folders
- Windows Additions: fixed a potential crash in the WDDM driver with Windows 10 (bug #14190)
- Solaris Additions: added support for X.Org Server 1.17
- X11 Additions: various seamless mode fixes, including invisible windows under LXDE.

15.25 Version 5.0.0 (2015-07-09)

This is a major update. The following major new features were added:

- Paravirtualization support for Windows and Linux guests to improve time-keeping accuracy and performance (see chapter 10.4, *Paravirtualization providers*, page 240)
- Make more instruction set extensions available to the guest when running with hardware-assisted virtualization and nested paging. Among others this includes: SSE 4.1, SSE4.2, AVX, AVX-2, AES-NI, *POPCNT*, *RDRAND* and *RDSEED*
- xHCI Controller to support USB 3 devices (see chapter 3.10.1, *USB settings*, page 56)
- Drag and drop support (bidirectional) for Windows, Linux and Solaris guests
- Disk image encryption (see chapter 9.31, *Encryption of disk images*, page 226)
- VMs can now be started in *separate mode*. The VM process is started headless while the frontend runs as a separate process which can be terminated without stopping the VM.
- GUI: VM guest-content scaling support (including 3D acceleration)
- GUI: New User Interface settings page for customizing status-bar, menu-bar and guest-content scaling
- GUI: New Encryption settings tab for customizing encryption options for disk images
- GUI: HiDPI support including application icons and optional unscaled HiDPI output on Mac OS X (including 3D acceleration)

- GUI: Hotplugging support for SATA disks
- New, modular audio architecture for providing a better abstraction of the host audio backends
- Support for the NDIS6 networking framework on Windows (default on Vista and later)

In addition, the following items were fixed and/or added:

- VMM: improved timing on Solaris hosts with older VT-x hosts without preemption timers
- VMM: further improvements for TSC frequency measurements and guest timekeeping
- VMM: debug facility now includes the guest CPU's FPU/SSE/extended state in the core dump
- VMM: fixed a hang under rare conditions on 32-bit hosts
- VMM: several fixes
- GUI: improved HID LEDs synchronization for Mac and Windows hosts. The physical LEDs state now restored together with the VM state.
- GUI: take the guest screen aspect ratio into account for the preview window
- GUI: provide direct access to storage media in the VM selector
- GUI: allow to save the VM state from the selector even if the VM is already paused
- VBoxManage: when exporting an appliance, support the suppression of MAC addresses, which means they will be always recreated on import, avoiding duplicate MAC addresses for VMs which are imported several times
- VBoxManage: now supports renaming storage controllers and USB controllers
- Guest Control: major overhaul, for example fixing wrong parameter quoting (bug #13157)
- USB: added USB traffic capturing (see chapter 9.29, *Capturing USB traffic for selected devices*, page 226)
- Made resizing X11 guests work more reliably
- API: block the removal of the current snapshot if it has child snapshots (only relevant for VMs without snapshottable hard disks, their presence always prevented removal), which resulted in VM config corruption
- API: mark VM configs with snapshots but without current snapshot as inaccessible, as this combination is nonsense
- API: fix information for some automatically generated events (only with XPCOM, Windows host was not affected), which caused errors when getting some of the attributes over the webservice (bug #12379)
- API: fix crashes in Java API clients using the XPCOM binding, happened with output parameters only (bug #11232)
- API: a number of settings (e.g. network settings) can now also be changed when the VM is in saved state
- API: fixed incorrect resuming of VMs on host-resume unless they were previously paused due to a host-suspend

- API: don't lose the saved state and "current state changed" flag during cloning of a VM
- API: OS type description consistency fix (bug #14162)
- VBoxSVC: don't keep the support driver permanently open
- Main/Properties: properly drop transient guest properties when the VM is powered off
- VRDP: fixed a couple of races which may cause a crash during VM poweroff
- ExtPack: don't fail if the TMP directory contains non-latin1 characters (bug #14159)
- 3D: fix potential race in which might cause a crash on VM termination
- 3D: fixed a possible memory leak in the host service
- Serial: new TCP/IP backend (see chapter 3.9, *Serial ports*, page 55)
- Storage: added USB mass storage device class (see chapter 5.1, *Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe*, page 79)
- Storage: added *vbox-img* standalone tool for direct manipulation of virtual hard disk images without VBoxManage
- Storage: fixed crash as a result of I/O errors in certain conditions (bug #13105)
- NAT: fixed several potential crashes
- NAT: don't forcibly reset/drop all connections when the link goes down
- Netsniffer: properly handle changing of the trace file name at VM runtime
- Audio: fixed audio output and input when changing the default audio device more than once on OS X
- Audio: fixed audio input on OS X under certain circumstances
- ICH9: fixed the interrupt disable logic for MSI interrupts; should fix old Linux guests with AHCI
- USB: improve playback with USB sound devices attached to the emulated OHCI controller
- Audio: provide Linux guests a different AC'97 audio codec type so Linux ALSA does not mis-detect the link speed (default for new VMs)
- BIOS: fix for booting from SCSI CD/DVD media
- BIOS: fix for reads partially beyond end of disk (bug #14021)
- VRDP: fixed listening for IPv6 on some systems (bug #14038)
- rdesktop-vrdp: upgraded to version 1.8.3
- Linux hosts: fixed a bug which made the netfilter driver ignore certain events (bug #12264)
- Mac OS X hosts: El Capitan fixes
- Mac OS X hosts: fixed a bug which might trigger a host kernel panic if a VM is started and another hypervisor is active
- Solaris hosts: Solaris 12 installer fix

- Guest Additions: added a heartbeat service (see chapter 9.30, *Configuring the heartbeat service*, page 226)
- Linux hosts / guests: support for Linux distributions using systemd without sysv emulation (e.g. ArchLinux)
- Windows Additions/WDDM: improved video memory utilization and allow more/bigger guest screens with large resolutions (including HiDPI)
- Linux Additions: added *-s* parameter to *mount.vboxsf* to be sloppy with invalid parameters
- X11 Additions: fixed wrong DPI value (bug #14151)
- Mac OS X guests: limit the CPU family for legacy guests
- Solaris Additions: added quiesce support to co-operate with Solaris' fast-reboot feature

15.26 Version 4.3.28 (2015-05-13)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed a Guru Meditation when rebooting certain guests (for example Solaris doing fast reboot) by fixing the implementation for INIT IPI
- VMM: added some information for diagnosing rare *VERR_VMX_INVALID_VMXON_PTR* Guru Meditations (VT-x only)
- GUI: HID LEDs sync: prevent synchronization if VM window has no focus (Windows and Mac OS X hosts only)
- GUI: fixed drag and drop moving the cursor between guest screens on certain hosts
- 3D: fixed a crash on restoring the VM state on X11 hosts (bug #12737)
- 3D: fixed a crash on restoring the VM state
- 3D: fixed a crash on Linux guest shutdown (bug #12772)
- VRDP: fixed incompatibility with rdesktop 1.8.3
- VRDP: fixed listening for IPv6 on some systems (bug #14038)
- Storage: don't crash if creating an asynchronous I/O context fails (e.g. when starting many VMs) and show a proper error message
- Floppy: several fixes
- Audio: improved the behavior of the volume control for the HD audio device emulation
- USB: increase the number of supported drivers from 3 to 5 (Windows hosts only)
- PS/2 keyboard: synchronize the LED state on VM restore (Windows and Mac OS X hosts only)
- NAT Network: when running multiple NAT networks with multiple VMs, only stop the respective services when stopping VMs (bug #14090)
- NAT: don't kill UDP bindings on ICMP errors (bug #13475)
- NAT: bandwidth limit now works properly with NAT (bug #11485)

- BIOS: fixed the returned size value of the VBE 2.0 PMI function 0Ah (4.2.0 regression; bug #14096)
- Guest Control: fixed parameter quoting in Windows guests (bug #13157)
- Webcam passthrough improvements for Linux (V4L2) hosts to support more webcam models
- API: don't fail starting a VM with *VBOX_E_INVALID_OBJECT_STATE* under certain conditions (bug #13617)
- API: be more verbose on *VBOX_E_INVALID_OBJECT_STATE* if a medium is attached to a running VM (bug #13560)
- API: fixed a bug which could result in losing certain screen resize events with multi-monitor guests
- rdesktop-vrdp: fixed path to the keymaps (bug #12066)
- rdesktop-vrdp: switch to version 1.8.3
- Windows hosts: more hardening fixes (e.g. bugs #14051, #14052)
- Linux hosts: another fix for activated SMAP on Linux 3.19 and newer (Broadwell and later; bug #13961)
- Linux hosts: Linux 4.1 compile fix (bug #14081)
- Solaris hosts: fixed using of VNIC templates with Crossbow based bridged networking to be compatible with vanity interface names
- Mac OS X hosts: fixed crash during VM termination under rare circumstances
- Windows Additions/WDDM: improved video memory utilization and allow more/bigger guest screens with large resolutions (including HiDPI)
- X11 Additions: prevent flickering when updating mouse cursor
- Solaris Additions: fixed incorrect usage of 'prtconf' while installing Guest Additions (Solaris 10 only)

15.27 Version 4.3.26 (2015-03-16)

This is a maintenance release. The following items were fixed and/or added:

- GUI: in the snapshots pane, protect the age of snapshots against wrong host time (bug #13955)
- NAT Network: fixed a bug which prevented to propagate any DNS name server / domain / search string information to the NAT network (4.3.24 regression; bugs #13915, #13918)
- NAT Network: don't delay the shutdown of VBoxSVC on Windows hosts
- Mouse support: the mouse could not be moved under rare conditions if no Guest Additions are installed (4.3.24 regression; bug #13935)
- Storage: if the guest ejects a virtual CD/DVD medium, make the change permanent (bugs #9858, #12885)
- VGA: made saving secondary screen sizes possible in X11 guests

- SDK: fixed the VirtualBox.tlb file (4.3.20 regression; bug #13943)
- rdesktop-vrdp: make it work with USB devices again (4.3.14 regression; bug #13901)
- USB: fixed a possible BSOD on Windows hosts under rare conditions
- iPXE: enable the HTTP download protocol on non-Linux hosts (bug #13628)
- Mac OS X hosts: don't panic on hosts with activated SMAP (Broadwell and later; bug #13951)
- Linux hosts: don't crash Linux 4.0 hosts (bug #13835)

15.28 Version 4.3.24 (2015-03-02)

This is a maintenance release. The following items were fixed and/or added:

- VMM: emulation fix for the *ENTER* instruction under certain conditions; fixes Solaris 10 guests (VT-x without unrestricted guest execution)
- VMM: fix for handling NMIs on Linux hosts with X2APIC enabled
- NAT/NAT Network: fix connection drops when the host's DHCP lease was renewed (4.3.22 regression; Windows hosts only; bug #13839)
- NAT: don't crash on an empty domain list when switching the DNS host configuration (4.3.22 regression; Mac OS X hosts only; bug #13874)
- PXE: re-enable it on Windows hosts (4.3.22 regression; Windows hosts only; bug #13842)
- Shared Folders: fixed a problem with Windows guests (4.3.22 regression; bug #13786)
- Audio: improved record quality when using the DirectSound audio backend
- VBoxManage: when executing the *controlvm* command take care that the corresponding VM runtime changes are saved permanently (bug #13892)
- Windows Installer: properly install the 32-bit version of VBoxRes.dll on 32-bit hosts (bug #13876)
- Linux hosts / guests: Linux 4.0 fixes (bug #13835)
- OS/2 Additions: fixed mouse integration (4.3.22 regression; bug #13825)

15.29 Version 4.3.22 (2015-02-12)

This is a maintenance release. The following items were fixed and/or added:

- VMM: refined measurement of TSC frequency on the host, improves timekeeping for guests
- VMM: decreased CPU load resulting from guest MMIO writes to the virtual APIC
- VMM: fixed interception of debug exceptions, observed while using the dbx debugger on Solaris guests (VT-x only)
- GUI: 3D overlay window positioning code improved, fixed potential misplacement of 3D accelerated guest graphics content
- GUI: fixed accident SSL authentication failures during update check on Windows hosts (bug #12969)

- GUI: never send the “ACPI power” keyboard scancode to the guest, we have the ACPI power button for that
- GUI: was unable to properly restore seamless mode VM from snapshot/saved-state under some circumstances
- VBoxHeadless: don’t crash if 3D is enabled in the VM settings (bug #10250)
- ATA: fixed several passthrough issues (bugs #12310, #1360)
- Audio: fixed DirectSound failure when the the host has no audio input device (Windows hosts only; bug #9205)
- SB16: fixed compatibility issue (bug #13769)
- Storage: fixed broken CD/DVD passthrough when using the IDE controller (bug #12310)
- NAT: new ping proxy for Windows hosts (bug #11871)
- NAT: Properly report outbound connect(2) failures to guest with TCP RST or ICMP (bug #10525)
- NAT Network: no need for frequent wakeups in VBoxNetDHCP and VBoxNetNAT (bug #11681)
- Host-only adapter: prevent Windows from creating an “Unidentified network” (bug #9688)
- Bridged Networking: don’t leak host-to-guest traffic to the wireless network when bridging to a wireless interface (bug #13714)
- Main: fixed a possible race when changing the medium leading to a deadlock under rare conditions (bug #13722)
- VBoxManage: fixed return code if starting a VM failed (bug #13773)
- Settings: on Windows host, do not use environment variable HOME at all, the settings location is derived from the user profile directory (bug #7689)
- API: fixed 2 deadlock opportunities related to medium handling (bugs #13789, #13801, thank you Alexander Urakov)
- API: fixed bug in XPCOM which created too few worker threads, sporadically resulting in a deadlock (bug #13802, thank you Alexander Urakov)
- SDK: fixed a garbage collection leak in the Python VirtualBox webservice API binding (bug #13817)
- Linux hosts: fixes for activated SMAP (Broadwell and later, bug #13820)
- X11 guests: prevent unwanted hiding of guest screens on multi-monitor guests (bug #13287)
- X11 guests: added support for X.Org Server 1.17
- X11 Additions: fixed a memory leak in VBoxService if libdbus is available but dbus-daemon isn’t running (bug #13770)
- Windows Additions: prevent VBox WDDM driver from loading if host reports weak OpenGL capabilities. 3D content now can be shown over Remote Desktop connection.
- Winodws Additions: some fixes for recent Windows 10 Previews

- Linux Additions: fixed a compatibility issue with 64-bit Linux 2.4 kernels
- Linux Additions: fixed a potential use-after-free when unloading the VBoxGuest module
- Linux Additions: Linux 3.19 fixes (bug #13741)

15.30 Version 4.3.20 (2014-11-21)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed reboot hang of 32-bit Windows SMP guests (bugs #13319, #13462)
- VMM: proper *Math Fault* handling with certain legacy guests (bug #9042, AMD hosts)
- VMM: fixed a Guru Meditation *VINF_EM_TRIPLE_FAULT* on older CPUs that don't support MSR bitmaps (VT-x only; bugs #13034, #13125, #13311, #13425, #13426, #13463, #13585)
- GUI: fix 3D overlay window reparenting issue when VM goes to full-screen mode on X11 hosts
- GUI: fix occasional loss of focus in full-screen mode on X11 host systems (4.3.16 regression)
- GUI: Mac OS X: wizards should have Cancel button (bug #12541)
- GUI: added a global option to prevent automatic raising of the new window by mouse move with multi-screen guests (bug #8878)
- API: accept remote display port 0 as the default RDP port (bug #8534)
- VBoxManage: fixed crash when executing *showvminfo* command under certain circumstances (bug #13190)
- ACPI: fixed occassional Guru Meditations in ACPI timer code (4.3.18 regression; bug #13521)
- EFI: improved performance of IDE disk access
- EFI: fixed a bug in the EFI video driver which prevented Windows to boot in UEFI mode (bug #12022)
- EFI: properly announce the amount of RAM for big VMs (bugs #11103 and #13211)
- Storage: fixed a crash under certain cicrumstances when a medium was ejected from a drive attached to the SATA controller without inserting a new medium before pausing or closing the VM (4.3.16 regression)
- Storage: fixed an interrupt acknowledge issue causing hanging guests or slower I/O (4.3.18 regression)
- Storage: fixed broken resume after the VM was suspended due to a full disk if host I/O caching is used
- Storage: fixed a Guru Meditation under certain conditions when using the DevLsiLogic controller with VMs running in software virtualization mode (4.3 regression; bugs #12254, #12655, #12709, #12774, #12886)
- Guest Control: fixed a bug which might lead to a crash during recursive copy
- SDK: Java COM bindings fixes

- iPXE: enable the HTTP download protocol (bug #13628)
- Runtime: do not use a fixed stack size creating temporary threads during initialization (bug #13038)
- Windows hosts: fixed more startup problems on certain Windows hosts due to conflicts with anti-virus software; better error reporting (4.3.14 regression; bug #13187)
- Windows hosts: fixed DirectSound host audio failure under certain conditions (bug #13418)
- Windows hosts: fixed additional cases of 4.3.14 regression whereby AltGr stopped working for some people (bug #13216)
- Windows Additions: preserve guest monitor layout when resizing Windows 7 or newer guests
- Linux Additions: Linux 3.18 compile fixes (bug #13515)

15.31 Version 4.3.18 (2014-10-10)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed a potential misbehavior after restoring the A20 state from a saved state
- GUI: fixed full-screen mode mini-toolbar related regressions for different platforms and window managers (bug #13369)
- GUI: X11: fixed full-screen mode Unity panels quirk caused by mini-toolbar code changes in last release (bug #13365)
- GUI: X11: added possibility to use legacy full-screen mode as the new one can cause multi-screen issues under Unity, see chapter 9.20.13, *Requesting legacy full-screen mode*, page 218 (bug #13365)
- GUI: Mac OS X: fixed full-screen mode artifact causing black screen when 3D acceleration was enabled on 10.10 Yosemite hosts (bug #13448)
- GUI: Mac OS X: fixed regression in user-space swiping from/to VBox in full-screen mode
- GUI: Mac OS X: fixed issue with switching to VBox in full-screen mode through Alt+Tab and Mission Control
- Storage: fixed data corruption when resizing huge VHD images under certain circumstances (bug #11960)
- Storage: fixed a rare hang during startup when the BIOS enumerates the storage devices attached to the SATA controller
- Storage: follow the spec with AHCI interrupt acknowledge (bug #13474)
- Storage: fixed broken iSCSI authentication (4.3.14 regression; bugs #13386, #13435)
- NAT Network: properly parse port forwarding rules to allow UDP rules
- USB: fixed a crash on Linux hosts with older Linux kernels (bug #13400) and several other fixes
- ACPI: fixed ACPI timer anomalies (bug #12076)

- Guest Control: fixed a memory leak (bug #13434)
- Main: when removing a VM, do also remove the VBoxStartup.log file which might exist on Windows hosts (bug #13478)
- Windows hosts: fixed more startup problems on certain Windows hosts due to conflicts with anti-virus software; better error reporting (4.3.14 regression; bug #13187)
- Windows hosts: propagate the process startup information to the child process (4.3.14 regression; bug #13243)
- Mac OS X hosts: don't force using the discrete GPU (bug #11111)
- Windows Additions: some Windows 10 tweaks
- X11 guests: fix a bug handling video driver display properties which prevented GNOME Shell on Fedora 21 from starting
- Linux hosts / guests: fixed a few remaining warnings in the kernel log if memory allocation fails (bug #11171)

15.32 Version 4.3.16 (2014-09-09)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed restoring 32-bit FPU state on 64-bit capable VMs and restoring guest FPU in raw-mode VMs (bug #12646; 4.3 regression)
- GUI: properly restore normal/scale mode guest-screen size after exiting full-screen/seamless mode
- GUI: mini-toolbar should provoke less artifacts/conflicts with 3D guest rendering
- GUI: Mac OS X: Native full-screen multi-screen transition was able to blackout host-screens for nearly minute
- GUI: X11: Modern window managers should now use native full-screen multi-screen mapping API
- GUI: added extradata item for configuring the mouse capture behavior, see chapter 9.20.12, *Configuring automatic mouse capturing*, page 218 (bug #3506)
- Storage: fixed a VBoxSVC crash when querying an iSCSI target with authentication configured (4.3.14 regression)
- Storage: fixed a rare data corruption during reads if another allocating write is running concurrently and accesses the same range
- Storage: fixed a rare crash for certain VHD images from other products
- Storage: fixed a rare release assertion when using the AHCI controller
- Floppy: fixed read errors and guest memory corruption when running under control of QEMM
- 3D: added experimental support for rendering on offline GPUs for Mac OS X host
- 3D: fixed white window appearing on entering FullScreen mode on Mac OS X host
- 3D: fixed video recording support for 3D data regression (bug #13073)

- 3D: fixes for MS Office 2013 support
- 3D: several fixes
- Bridged Networking: improved IPv6 support when bridging to a wireless interface
- NAT: prevent internal DNS service from stuck in host-resolver mode when host was switched from one network to another one while host was sleeping (Mac OS X hosts)
- NAT: preserve DF (if possible) and TOS when proxying outbound UDP datagrams (bugs #9440, #12309)
- NAT: don't let multicast datagrams out (bug #7338)
- NAT: fixed handling of large incoming UDP datagrams on Windows hosts (bug #12136)
- NAT: fixed handling of the RFC 1533 DHCP PAD option
- NAT Network: fixed inbound half-close on Windows hosts
- NAT Network: preserve IPv4 DF (if possible), TTL, TOS and IPv6 Hop Limit when proxying outbound UDP datagrams
- VRDP: fixed a rare crash when using remote audio input
- USB: fixed several regressions from 4.3.14 (bug #13320)
- Audio: made the HDA sound emulation work with certain Mac OS X guests (e.g. Mountain Lion)
- Windows hosts: fixed startup problems on certain Windows hosts due to conflicts with anti-virus software (4.3.14 regression; bug #13187)
- Windows hosts: fixed 4.3.14 regression whereby AltGr stopped working for some people (bug #13216)
- X11 hosts: made the extra key on Brazilian Thinkpads work (bug #8745)
- X11 hosts: fixed a problem of input focus cycles and immediately released key presses in full screen mode (bug #13238)
- Linux hosts: fixed flooding the kernel log with USB related messages when passing through certain USB devices to a VM (bug #13085)
- Linux guests: stop applications crashing when *drm_wait_vblank* is called (bug #13265)
- Linux guests: fix a crash in gnome-session (bug #13335)
- X11 guests: do not start VBoxClient over an SSH connection (bug #13107)
- X11 guests: added support for X.Org Server 1.16 (bug #13207)
- X11 guests: fixed a wrong parameter in the video driver which caused problems with full-screen X11 clients (bug #2748)
- VirtualKD: introduced stub/loader device for speeding up Windows kernel debugging, details see `http://virtualkd.sysprogs.org/`

15.33 Version 4.3.14 (2014-07-15)

This is a maintenance release. The following items were fixed and/or added:

- VMM: more fixes for MSR emulation on certain hardware (bugs #12784, #12949, #13034)
- VMM: improve MSI handling under rare circumstances (only relevant for the ICH9 chipset)
- VMM: fixed #UD exception for 64-bit guests with the EFER.SCE bit and the *SYSCALL* instruction (VT-x only; 4.3 regression; bug #13008)
- VMM: fixed timekeeping after resuming SMP guests
- VMM: properly wake up a halted VCPU on NMI/SMI
- GUI: fixed a potential crash
- GUI: fixed stuck AltGr key on Windows hosts (bug #2537)
- GUI: fixed a potential error during the version check
- GUI: shortcut change should not require Enter/Return (or other trigger) to confirm (bugs #12828, #12847, #12937, #13087)
- GUI: fixed update check which was broken due to changing the location of the root certificates (bug #13096)
- VBoxManage: fixed typo in *showvminfo –machinereadable* (bug #13176)
- NAT: fixed inbound half-close (bug #13116)
- NAT: fixed slow upload speed under certain conditions (bug #10034)
- NAT Network: fixed potential loss of inbound TCP data
- NAT Network: fixed potential infinite stalls of TCP connections over IPv6
- NAT Network: fixed resets of TCP connections on Windows hosts
- NAT Network: fixed inbound half-close on Mac OS X hosts
- NAT Network: fixed socket leak on Solaris hosts
- NAT Network: fixed ping of mapped host loopback on Mac OS X and Solaris hosts, fixed proxying of IMCP errors on Mac OS X
- Host-Only Network: fixed SNMP *ifConnectorPresent* value on Windows (bug #13143)
- Storage: fixed a possible crash with CD/DVD passthrough under certain circumstances
- Storage: fixed a crash when trying to open an inaccessible QED or QCOW image (bug #12613)
- Storage: fixed data corruption or read errors under rare circumstances
- AHCI: fixed a crash under rare circumstances
- USB: performance fixes
- ICH9: properly reset MSI capability on reset
- Keyboard: active modifier keys during suspend were stuck after resuming the host

- 3D: fixed misbehavior with huge guests (i.e. guest more than 4GB guest memory
- 3D: several fixes
- API: properly detect the Windows 8.1 guest OS type (bug #13090)
- ExtPack: cleanup of dangling uninstallation directories
- Linux hosts / guests: compile fix for EL7 (bug #12638)
- Linux Additions: made 3D pass-through work with recent versions of Mesa in the guest (bug #12941)
- Linux Additions: Linux 3.16 fixes (bug #13123)
- Mac OS X hosts: when scanning for host CD/DVD devices also consider BlueRay devices
- Mac OS X hosts: fixed host shutdown and reboot delay caused by running VBoxSVC process in some cases
- OS/2 Additions: fixed gengradd.dll library name (bug #12785)
- Solaris Additions: fixed permissions of files and directories located on shared folders
- Windows host installer: fixed the need for rebooting Windows after installation or upgrade, extended logging for NetFlt/NetAdp (un)installation

15.34 Version 4.3.12 (2014-05-16)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed an occasional Guru Meditation (Mac OS X hosts only; bugs #12727, #12954)
- VMM: fixed a rare condition that would fail to invalidate guest TLB entries or would invalidate them when not required (Windows hosts only)
- VMM: fixed a *VERR_NOT_SUPPORTED* Guru Meditation seen with certain guests, e.g. OpenServer 5.0.7
- VMM: more fixes for MSR emulation on certain hardware (bugs #12240, #12875)
- GUI: fixed mouse positioning with mouse integration disabled and multiple guest screens (Windows hosts only; bug #9059)
- GUI: fixed crash in VM manager (bug #12878)
- GUI: fixed crash under rare conditions on entering/exiting full-screen/seamless mode
- Shared Clipboard: don't stop working after taking a snapshot (bug #12700)
- AHCI: fixed a crash under rare circumstances
- API: fixed a hang during VM shutdown under rare conditions
- NAT: fixed generation of malformed ICMP error datagrams (4.3.10 regression)
- NAT: fixed potential crash in DNS proxy
- NAT Network: don't drop port forwarding rules after some time
- NAT: fixed ARP cache corruption and network loss in Windows guest caused by iSCSI service activity

- USB: improved check if a storage device is currently mounted to the host when the device is about to be attached to the VM (Mac OS X hosts only; #11038)
- 3D support: several fixes, including better support for Ubuntu 14.04
- VRDP: fixed a potential crash on client disconnect (bug #12858)
- VBoxSVC: fixed a race when a new client is started a few seconds after the last client terminated (Windows hosts only; bugs #11309, #12509)
- VBoxSVC: fixed VirtualBox.xml registry corruption after VM renaming
- VBoxSVC: fixed a potential crash caused by incorrect USB device filter (Mac OS X hosts only; #11038)
- Windows hosts: partly support 32-bit COM on 64-bit systems
- Windows host installer: implemented merge module (msm) support
- Linux hosts: fixed dependency of boot script on older Debian systems (bug #12262)
- Linux guests: fix symbolic link to shared folder helper (bug #12879)
- Linux Additions: don't crash VBoxService during guest execute for users without a password (bug #12994)
- Linux Additions: fixed a bug in guest execution where the guest process terminated with *VERR_INTERRUPTED* to the host

15.35 Version 4.3.10 (2014-03-26)

This is a maintenance release. The following items were fixed and/or added:

- VMM: more work on improving the emulation of certain MSR registers on certain host CPUs (e.g. bugs #12734, #12736, #12744, #12748, #12686, #12770)
- VMM: fixed single-stepping for real-mode guests (VT-x without unrestricted guest execution) and some I/O instructions (bug #12636)
- VMM: fixed a potential problem with COW pages if nested paging is not available
- GUI: Mac OS X: experimental native full screen support for Mountain Lion and Mavericks (bug #12292)
- GUI: Mac OS X: removed the mini-toolbar minimize button which doesn't work under Mac OS X full screen mode anyway
- GUI: experimental HID LEDs synchronization for Windows and Mac OS X hosts: fixed keyboard re-synchronization if the feature is disabled (as done by default; bug #12758)
- GUI: fixed a potential crash when opening the preferences menu (bug #12862)
- OVF: fixed a crash of the VirtualBox Manager when re-starting guest export (bug #12586)
- 3D support: several fixes
- HGCM: fixed a problem with saved states which could cause several guest misbehavior after a VM was started from a saved state
- Storage: fixed a bug preventing to compact differential snapshots under certain conditions

- VBoxSVC: fixed a segmentation fault on Linux hosts if a very long path exists under /dev (bug #12760)
- API: fixed guest misbehavior under certain conditions if a storage medium was attached or removed at VM runtime
- Windows installer: make the *–silent* parameter work again (bug #12764)
- Mac OS X Networking: prevent local traffic (VM-to/from-host) from leaking to wire (bug #12750)
- Windows Additions: fixed the environment for guest processes (4.3.8 regression; bug #12782)
- Windows Additions/WDDM: fixed divide by zero exception with multiple guest screens under certain conditions
- Windows Additions/WDDM: fixed crashes with 2D video acceleration enabled (4.3.8 regression; bug #12745)
- Linux Additions: install correctly on Ubuntu guest systems with a /usr/lib64 directory (bug #12513)
- X11 Additions: fix for the VBoxClient process not exiting correctly (bug #12348) and consuming too much processor time

15.36 Version 4.3.8 (2014-02-25)

This is a maintenance release. The following items were fixed and/or added:

- VMM: more work on improving the emulation of certain MSR registers (e.g. bugs #12224, #12544)
- VMM: fixed a *VERR_INVALID_RPL* Guru Meditation when booting certain guests (bug #11350)
- VMM: experimental support for SSE 4.1 / SSE 4.2 passthrough, see the user manual how to enable it (bug #8651)
- VMM: fix for recent Linux kernels with software virtualization
- GUI: experimental HID LEDs synchronization for Windows hosts, see chapter 9.28, *Support for keyboard indicators synchronization*, page 225
- GUI: warn the user if the Oracle Extension Pack is not installed and the user tries to activate the remote display feature (bug #9104)
- GUI: make sure that a minimized guest (using mini toolbar in full screen / seamless mode) keeps the minimized state (bug #12199)
- GUI: popup banner's "do not show this message again" check-box replaced with corresponding button
- GUI: network adapter cables can now be connected/disconnected directly through the running virtual machine Devices / Network menu a Network status-bar indicator
- GUI: the new VM wizard now proposes 64-bit guests on 64-bit hosts by default; better distinction between 32-bit OS types 64-bit OS types (bug #12533)

- GUI: better error message if appliance import fails (bug #12657)
- GUI: allow to set host-combination to 'None' using the Global settings / Input page (bug #12730)
- GUI: don't switch the guest to a black screen during online snapshot merge (4.3 regression)
- VBoxManage: when exporting an appliance, support the suppression of MAC addresses, which means they will be always recreated on import, avoiding duplicate MAC addresses for VMs which are imported several times
- AHCI: fixed a VM hang during suspend under certain circumstances
- AHCI: fixed a VM hang during online snapshot merge under certain circumstances
- AHCI: fixed a bug which resulted in Windows XP guest hangs if a SATA CDROM is attached (bug #12417)
- AHCI: fixed a Guru Meditation under certain conditions
- AHCI: ejecting a CD/DVD medium failed under certain conditions
- AHCI: disk hotplugging fixes
- NAT: transparent handling of host sleep/resume and network configuration changes if the dnsproxy is enabled or if the hostresolver is used (bug #12441)
- NAT: fixed crash and misbehaviour under some circumstances with ICMP packets having TTL=1
- NAT Network: fixed IPv6 reassembly
- NAT Network: ping proxy implemented
- OVF: fixed reading of the OVF 0.9 section element (4.3 regression; bug #12345)
- OVF: several fixes
- 3D support: several fixes, multiscreen fixes (e.g. bug #9124)
- 3D support: include 3D content in captured videos (bug #12666)
- 3D support: include 3D content in captured screenshot (bug #11758)
- VGA: proper handling of legacy graphics modes if the Guest Additions are active (bug #6649)
- USB: fixed crash during isochronous transfer under rare circumstances
- BIOS: better disk geometry handling of SCSI drives
- API: fix crashes in Java API clients using the XPCOM binding, happened with output parameters only (bug #11232)
- VBoxSVC: documented the handling of host power management events (see chapter 9.26, *Handling of host power management events*, page 225) and added an extradata item for configuring the handling of the battery-low event (bug #9925)
- VBoxSVC: fixed a bug which could trigger a crash if a VM snapshot was restored the second time and the VM has associated bandwidth groups (bug #12569)
- VBoxSVC: properly detect ifconfig if located in */bin* (bug #12713)

- Shared Folders: fixed a failure to restore transient shared folders when starting a VM from a saved state (bug #12578)
- Mac OS X hosts: fixed issue when the application icon was frozen in the dock if the bridging interface was not connected to a network (bug #12241)
- Linux hosts: also consider the physical package ID when determining the number of physical CPU cores
- Linux hosts / guests: don't warn in kernel log if memory allocation fails (bug #11171)
- Solaris hosts: fixed the autostart SMF script (bug #11720)
- Windows hosts: fixes for non-ANSI code page user names and similar environment contents (bug #12596)
- Windows hosts / guests: fixed setting and using a guest user's process environment variables (relevant for Guest Control)
- Windows Additions: fixed handle leaks in VBoxTray (bug #12563)
- Windows Additions: fixed a crash while detecting active guest users
- Windows Additions: fixed restoring backed up D3D files on XPDM -> WDDM upgrade
- Guest Control: fixed setting and using a guest user's process environment variables
- Linux Additions: support Enterprise Linux 6.5 kernels (bug #12505)
- Linux Additions: fixed CPU hot-remove on newer Linux kernels
- Linux / Solaris Additions: don't automount a shared folder which is already mounted
- X11 Additions: support X.Org Server 1.15 (bug #12623)

15.37 Version 4.3.6 (2013-12-18)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed a Guru Meditation *VINF_EM_TRIPLE_FAULT* caused by VMCB caching with nested paging on certain AMD CPUs (bug #12451)
- VMM: fixed a Guru Meditation *VERR_VMX_UNEXPECTED_INTERRUPTION_EXIT_TYPE* while intercepting debug exceptions (VT-x only; bug #12410)
- VMM: fixed a Guru Meditation *VERR_SVM_UNEXPECTED_EXIT* while intercepting debug register accesses (AMD-V only; bug #12481)
- VMM: fixed a *VERR_SSM_STRUCTURE_MAGIC* error when trying to load a saved state made with VBox 4.3.4 when VT-x/AMD-V is disabled. Unfortunately, VBox 4.3.4 produced broken saved states for this configuration so you have to discard these states (bug #12414)
- VMM: added a few more MSRs to the whitelist required by certain guests (bug #12245)
- GUI: fixed deleting of inaccessible VMs (4.3 regression; bug #12205)
- GUI: fixed warnings in VM settings / number of guest processors (bug #12480)
- Main: don't automatically enable 64-bit guests on 64-bit hosts if VT-x/AMD-V is not available (bug #12424)

- Main: always expose the DMI memory information to Windows 2012 guests (bug #12017)
- Main: fixed occasional crashes on guest display resolution change (bug #7063)
- Main: fixed reporting back temporary name when calling *IGuestSession::DirectoryCreateTemp()* (bug #12498)
- API: fix for a hang when launching a GUI VM through the API, which crashes due to GUI unavailability
- Storage: fix for *BLKCACHE_IOERR* runtime errors under rare circumstances (bug #11030)
- Network: allow to start more than 5 PCNet instances (bug #12426)
- E1000: if the cable was disconnected before the guest initialized the device, the link status was not properly set to 'down' after the initialization completed despite the fact that there was no connection
- 3D support: fixed offset of guest 3D image elements (Mac OS X Retina hosts only; bug #11021)
- Solaris hosts: fixed accessing the host driver from non-global zones (4.3 regression; bug #12271)

15.38 Version 4.3.4 (2013-11-29)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fix for a bug in the Local APIC emulation causing a BSOD when booting certain guests (4.3.0 regression; bug #12240)
- VMM: fixed loading of saved states if VT-x/AMD-V was disabled (4.3.2 regression; bug #12291)
- VMM: fixed single-stepping inside the guest for certain instructions (VT-x only; bug #10947)
- VMM: fixed a performance issue involving APIC accesses after rebooting a VM (4.3.0 regression; VT-x only; bug #12296)
- VMM: fixed TPR patching to be enabled for 32-bit guests even when the chosen guest type is 64-bit (4.3.0 regression; AMD-V only)
- VMM: fixed occasional *VINF_EM_TRIPLE_FAULT* errors on hosts without the unrestricted guest execution feature (bug #12198)
- GUI: don't bother the user with the BPP warning if no Guest Additions are installed
- GUI: fixed machine-window paint artifacts on VM reboot / guest-screen resize
- GUI: make sure the assigned license and description are attached to the exported appliance
- GUI: fixed bugs in close VM action restrictions handling (bug #12333)
- GUI: fixed incorrect wizards text colors for some unusual look and feel styles (bug #11743)
- GUI: should restore seamless mode as soon as possible after VM reboot or shutdown
- GUI: fixes for medium enumeration

- GUI: the OS X hot corners were not accessible while a VirtualBox VM is running (Mac OS X hosts only; bug #4139)
- GUI: fixed an old bug which bared the host from cleanly shutdown / reboot if the VM selector window is open (Mac OS X hosts only; bug #8254)
- Host-only Network: fixed creating of host-only network interfaces (4.3.0 regression; bug #12182)
- NAT: don't run into an infinite loop in case the host cannot access any DNS server (4.3.0 regression; bug #12300)
- NAT: don't re-connect the cable if the DNS information changes and the cable was disconnected before (4.3.0 regression; bug #12225)
- NAT: fixed several issues with automatically starting / terminating of NAT networks on VM start / stop and configuration changes
- VBoxNetDHCP: don't block prevent VBoxSVC from terminating (bug #12264)
- 2D Video acceleration: fix crashes on presentation mode switches (bug #9194)
- BusLogic: allow to run VMs with more than one BusLogic SCSI controller enabled
- Keyboard: fixed a VM crash if a VM was resumed from a saved state where at least one key was pressed (bug #11289)
- VBoxSVC: fixed a heap corruption under certain conditions (4.3.0 regression)
- VBoxSVC: fixed a race leading to a hang during initialization (bug #12349)
- OVF: fixed import logic for OVF appliances containing multiple VMs
- OVF: improved logic for finding an appropriate image format during OVF import
- API: block the removal of the current snapshot if it has child snapshots (only relevant for VMs without snapshottable hard disks, their presence always prevented removal), which resulted in VM config corruption
- API: mark VM configs with snapshots but without current snapshot as inaccessible, as this combination is nonsense
- API: fixed information for some automatically generated events (only with XPCOM, Windows host was not affected), which caused errors when getting some of the attributes over the webservice (bug #12379)
- SDK: extended the functionality coverage for the C bindings
- Guest Control: various bugfixes and improved VBoxManage help (bugs #8072, #11044, #12336, #12338, #12346, #12371)
- Windows hosts: another attempt to fix the massive DPC latency (bug #6242)
- Windows host installer: make registering file extensions optional, contributed by Tal Aloni (bug #8009)
- Mac OS X hosts: properly sign the kernel extensions for Mavericks hosts (bug #12256)
- Mac OS X hosts: fixed a bug where the VirtualBox dock icon was not properly removed from the dock after a VM terminated preventing Mavericks hosts from shutting down (bug #12241)

- Mac OS X hosts: fixed minor installer issue (bug #12275)
- Linux hosts / guests: Linux 3.13 compile fixes (bug #12358)
- Linux guests: build the vboxvideo kernel module correctly on OL/RHEL 6.1 guests (bug #11996)
- Linux guests: make 3D work on Slackware 14.1 (bug #12320 comments 3 and 4)
- Guest Additions/3D: fixed an occasional dead-lock (bug #12319)
- Windows Additions/3D: fixed possible memory leaking (bug #12228)
- Windows Additions/XPDM: use separate tables containing valid video modes for each virtual monitor
- Windows Additions: fixed automatic logins for Vista and newer Windows guests (bug #12332)

15.39 Version 4.3.2 (2013-11-01)

This is a maintenance release. The following items were fixed and/or added:

- VMM: fixed restoring of the auxiliary TSC MSR in VT-x that caused host BSODs on Windows 8.1 hosts and unpredictable behavior on other hosts (bug #12237)
- VMM: provide fake values for a couple of MSRs to make more guests happy on certain hosts
- VMM: fixed detection of VT-x on certain machines where the BIOS would not set the VMX LOCK feature bit, which affected the VM settings in the GUI
- VMM: fixed TPR threshold which caused BSODs on Windows XP guests that use the I/O APIC (VT-x only; bug #12227)
- VMM: fixed PATM saved state incompatibility for software virtualized VMs (bug #12222)
- VMM: don't fail if AMD-V isn't available if the VM is configured to use software virtualization
- GUI: fixed guest resize breakage on visual representation mode change (when switching from normal to full screen etc)
- GUI: make sure the guest screen is resized after restoring a VM from a saved state if the host screen size changed
- GUI: disabled SCROLL LED sync from HID LEDs synchronization (Mac OS X hosts only)
- Webcam passthrough improvements including GUI support (see chapter 9.7.1, *Using a host webcam in the guest*, page 195)
- Guest Control: implemented more `IGuestSession` methods
- Guest Control: added support for deleting and renaming guest files + directories in VBoxManage
- Guest Control: various bugfixes
- API: incorrect handling of hardware UUID default value, resulting in an all zero DMI/SMBIOS UUID, which leads to Windows requesting re-activation (4.3 regression; bug #12244)

- 3D support: fixed crash on shutdown if 2D video acceleration is enabled (Mac OS X hosts only)
- 3D support: miscellaneous fixes
- Storage: fixed detection of CD/DVD media when switching from an empty to a host drive with passthrough enabled
- Storage: fixed hang of the VM process when the disk is full under certain circumstances
- NAT: listen for changes of NAT Network setting at runtime
- NAT: NAT Network DHCP server now saves leases to a persistent storage
- Main: monitor changes in host DNS configuration
- Mac OS X host: reworked a mechanism of adding a VM desktop alias from the VM selector
- Mac OS X installer: remove old kernel extensions during upgrade (bug #12258)
- Linux Additions: correctly set umask before installing (bug #12166)
- X11 Additions/3D: fix freezes starting 3D desktop (bug #11503, thank you Sam Spilsbury)
- X11 Additions/3D: fix depth buffer support (bug #11905)
- X11 Additions/3D: fix Age Of Empires 3 rendering (bug #11331)
- Windows Additions/3D: fix Google Earth plugin rendering
- Windows Additions/WDDM: autoresize fixes

15.40 Version 4.3.0 (2013-10-15)

This is a major update. The following major new features were added:

- VMM: major rewrite of the VT-x code and the AMD-V code including many bug fixes and performance improvements (for example bug #9659)
- VMM: introduced a lightweight instruction interpreter for situations not handled by hardware virtualization
- GUI: extended messaging mechanism (new non-modal popup overlays used to show non-critical warnings and provide user with additional information)
- GUI: keyboard shortcuts management (input page of global preferences extended with possibility to edit general keyboard shortcuts for VirtualBox Manager and Virtual Machine)
- GUI: video capturing support (bug #4766)
- Added USB touch device emulation
- Added experimental support for webcam passthrough complementing USB passthrough (see chapter 9.7.1, *Using a host webcam in the guest*, page 195)
- Added SCSI CD-ROM emulation, including boot support
- VRDP: support for IPv6
- Guest Control: guest sessions now are running in dedicated, impersonated session processes (needs at least Guest Additions 4.3 installed)

- Guest Control: implemented `IGuestFile` support
- NAT: experimental virtual router mode: several VMs are attached to the same internal network and share one NAT service (see chapter 6.4, *Network Address Translation Service*, page 96)

In addition, the following items were fixed and/or added:

- VMM: significantly improved performance of NetWare 5.x/6.x guests on host systems without nested paging support
- VMM: fixed losing host NMIs while in VT-x guest-context
- VMM: changed order of actions in emulated task switch (bug #10532)
- VMM: allow to activate VT-x while in SMX mode and provide more information if that is not possible
- GUI: update check uses https
- GUI: numerous minor internal cleanups and bug fixes
- GUI: HID LEDs synchronization when switching between guest window(s) and host (Mac OS X hosts only)
- GUI, VBoxManage: when unregistering a VM, also unregister the hard disk images which are used exclusively (bug #10311)
- GUI: use the number of physical presented processor cores instead of the number of logical processor cores to check if the users assigned too many virtual CPUs to the guest
- Snapshots: made live snapshots work again (bug #9255)
- Teleportation: made it work again (bug #9455)
- Storage: implemented AHA-154x compatibility mode in the emulated BusLogic SCSI HBA
- Storage: significantly improved performance of large ATAPI PIO transfers (BeOS, Minix 3 guests affected)
- Storage: added floppy formatting emulation (NB: cannot be used to change existing media geometry)
- Settings: global and per-VM default frontend configuration, useful to select the use of alternative VM frontends
- Settings: limit depth of snapshot tree to 250 levels, as more will lead to decreased performance and may trigger crashes
- Settings: the per-VM hwvirtextexcl setting has been replaced by a global hwvirtexclusive property
- Main: new event queue implementation which does not use the host's native event queue for processing VirtualBox events anymore
- Main: eliminate the use of SysV semaphores on all host OSes other than Windows, namely Linux, Solaris and Mac OS X, with the consequence that no system reconfiguration is needed to run more than approximately 100 VMs
- Main: use the XDG standard configuration folder instead of .VirtualBox on systems where it is appropriate (bug #5099)

- Main: extension pack framework can now support loading HGCM modules, contributed by Jeff Westphal
- VBoxManage: list more information about hard disk/DVD/floppy media, and support the `--long` option to show really all available details
- VBoxManage: added support for optional command line parameters for the automatic Guest Additions update
- VBoxManage: added support for listing active guest sessions, guest processes and/or guest files via `guestcontrol list <all|sessions|processes|files>`
- VBoxManage: added support for closing active guest sessions via `guestcontrol session close --session-id <ID>| --session-name <name or pattern>|--all`
- VBoxManage: added support for terminating active guest processes via `guestcontrol process kill|close|terminate --session-id <ID>| --session-name <name or pattern> <PID> ... <PID n>` or `guestcontrol [p[s]]kill --session-id <ID>| --session-name <name or pattern> <PID> ... <PID n>`
- VBoxManage: added support for watching guest sessions via `guestcontrol watch`
- VBoxManage: added `modifyvm --triplefaultreset` to make the VM reset on triple fault instead of triggering a Guru Meditation (see chapter 8.8, *VBoxManage modifyvm*, page 128)
- 3D support: several fixes
- 3D support: several fixes for Mac OS X hosts
- OVF: several fixes
- Extpack Installer: make it work if the file is located in a folder with special characters
- Keyboard: fix for reporting key sequences like Ctrl+Alt+Del for the USB keyboard emulation
- Shared Clipboard/X11: support for BMP-format images, contributed by François Revol
- Mac OS X hosts: limited support for Mac OS X 10.9 (Mavericks)
- Mac OS X hosts: use a launchd script instead of the deprecated StartupItem mechanism (bug #8940)
- Windows hosts: don't cause massive DPC latency (only on certain hosts; still needs improving; bug #6242)
- Windows hosts: consider symlinks when retrieving volume information (bug #11962)
- Windows hosts: fixed an issue with USB2 devices being inaccessible when plugged into USB 3.0 ports
- Windows Additions: fixed misbehavior with guest display power management (WDDM driver only; bug #11170)
- Windows Additions: fixed memory leak caused by WTSQuerySessionInformation() on Windows 2000 guests (bug #12072)
- Windows Additions: ability to track guest user idle times through the newly introduced event *IGuestUserStateChangedEvent*
- Linux Additions: fixed udev detection in the init script with Linux 3.x kernels

15.41 Older Change log details

With VirtualBox 5.0, changelog information for versions before 4.3 was removed in order to save space. To access this information, please consult the User Manual of VirtualBox version 4.3 or earlier.

16 Third-party materials and 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.

The source code for the materials listed below as well as the rest of the VirtualBox code which is released as open source are available at `http://www.virtualbox.org`, both as tarballs for particular releases and as a live SVN repository.

16.1 Materials

- VirtualBox contains portions of QEMU which is governed by the licenses in chapter 16.2.5, *X Consortium License (X11)*, page 336 and chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 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 the license in chapter 16.2.5, *X Consortium License (X11)*, page 336 and

  Copyright 2004 by the Massachusetts Institute of Technology.

- VirtualBox contains code of the BOCHS VGA BIOS which is governed by the license in chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 and

  Copyright (C) 2001, 2002 the LGPL VGABios developers Team.

- VirtualBox contains code of the BOCHS ROM BIOS which is governed by the license in chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 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 the license in chapter 16.2.6, *zlib license*, page 336 and

  Copyright (C) 1995-2003 Jean-loup Gailly and Mark Adler.

- VirtualBox may contain OpenSSL which is governed by the license in chapter 16.2.7, *OpenSSL license*, page 336 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 the license in chapter 16.2.3, *Mozilla Public License (MPL)*, page 330 and

  Copyright (C) The Authors.

- VirtualBox contains Slirp which is governed by the license in chapter 16.2.8, *Slirp license*, page 337 and was written by Danny Gasparovski.

  Copyright (C) 1995, 1996 All Rights Reserved.

- VirtualBox contains liblzf which is governed by the license in chapter 16.2.9, *liblzf license*, page 338 and

  Copyright (C) 2000-2005 Marc Alexander Lehmann <schmorp@schmorp.de>

- VirtualBox may ship with a modified copy of rdesktop which is governed by the license in chapter 16.2.1, *GNU General Public License (GPL)*, page 321 and

  Copyright (C) Matthew Chapman and others.

- VirtualBox may ship with a copy of kchmviewer which is governed by the license in chapter 16.2.1, *GNU General Public License (GPL)*, page 321 and

  Copyright (C) George Yunaev and others.

- VirtualBox may contain Etherboot which is governed by the license in chapter 16.2.1, *GNU General Public License (GPL)*, page 321 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 iPXE which is governed by the license in chapter 16.2.1, *GNU General Public License (GPL)*, page 321 and

  Copyright (C) Michael Brown <mbrown@fensystems.co.uk> and others.

- VirtualBox contains code from Wine which is governed by the license in chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 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 the license in chapter 16.2.11, *lwIP license*, page 338 and

  Copyright (C) 2001, 2002 Swedish Institute of Computer Science.

- VirtualBox contains libxml which is governed by the license in chapter 16.2.12, *libxml license*, page 339 and

  Copyright (C) 1998-2003 Daniel Veillard.

- VirtualBox contains libxslt which is governed by the license in chapter 16.2.13, *libxslt licenses*, page 339 and

  Copyright (C) 2001-2002 Daniel Veillard and Copyright (C) 2001-2002 Thomas Broyer, Charlie Bozeman and Daniel Veillard.

- VirtualBox contains code from the gSOAP XML web services tools, which are licensed under the license in chapter 16.2.14, *gSOAP Public License Version 1.3a*, page 340 and

  Copyright (C) 2000-2007, Robert van Engelen, Genivia Inc., and others.

- VirtualBox ships with the application tunctl (shipped as VBoxTunctl) from the User-mode Linux suite which is governed by the license in chapter 16.2.1, *GNU General Public License (GPL)*, page 321 and

  Copyright (C) 2002 Jeff Dike.

- VirtualBox contains code from Chromium, an OpenGL implementation, which is goverened by the licenses in chapter 16.2.15, *Chromium licenses*, page 345 and

  Copyright (C) Stanford University, The Regents of the University of California, Red Hat, and others.

- VirtualBox contains libcurl which is governed by the license in chapter 16.2.16, *curl license*, page 347 and

  Copyright (C) 1996-2009, Daniel Stenberg.

- VirtualBox contains dnsproxy which is governed by the license in chapter 16.2.4, *MIT License*, page 336 and

  Copyright (c) 2003, 2004, 2005 Armin Wolfermann.

- VirtualBox may contain iniparser which is governed by the license in chapter 16.2.4, *MIT License*, page 336 and

  Copyright (c) 2000-2008 by Nicolas Devillard.

- VirtualBox contains some code from libgd which is governed by the license in chapter 16.2.17, *libgd license*, page 347 and

  Copyright 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007 Pierre-Alain Joye (pierre@libgd.org).

- VirtualBox contains code from the EFI Development Kit II which is governed by the license in chapter 16.2.18, *BSD license from Intel*, page 348 and

  Copyright (c) 2004-2008, Intel Corporation.

- VirtualBox contains libjpeg which is governed by the license in chapter 16.2.19, *libjpeg License*, page 348 and

  Copyright (C) 1991-2010, Thomas G. Lane, Guido Vollbeding.

- VirtualBox may contain x86 SIMD extension for IJG JPEG library which is governed by the license in chapter 16.2.20, *x86 SIMD extension for IJG JPEG library license*, page 349 and

  Copyright 2009 Pierre Ossman <ossman@cendio.se> for Cendio AB; Copyright 2010 D. R. Commander; Copyright (C) 1999-2006, MIYASAKA Masaru.

- VirtualBox may ship a copy of Qt which is governed by the license in chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 and

  Copyright (C) 2010, 2011 Nokia Corporation and/or its subsidiary(-ies).

- VirtualBox contains parts of the FreeBSD kernel which is governed by the license in chapter 16.2.21, *FreeBSD license*, page 349.

- VirtualBox contains parts of the NetBSD kernel which is governed by the license in chapter 16.2.22, *NetBSD license*, page 350.

- VirtualBox contains portions of liblightdm-gobject which is governed by the license in chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 and

  Copyright (C) 2010-2013 Canonical Ltd.; Copyright (C) 2010-2011 Robert Ancell.

- VirtualBox contains portions of glib which is governed by the license in chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 and

  Copyright (C) 1995-2011 The Glib team

- VirtualBox contains portions of PCRE which is governed by the license in chapter 16.2.23, *PCRE license*, page 350 and

  Copyright (c) 1997-2012 University of Cambridge; Copyright(c) 2009-2012 Zoltan Herczeg; Copyright (c) 2007-2012, Google Inc.

- VirtualBox contains portions of libffi which is governed by the license in chapter 16.2.24, *libffi license*, page 351 and

  Copyright (c) 1996-2012 Anthony Green, Red Hat, Inc and others. See source files for details.

- VirtualBox contains portions of FLTK which is governed by the licenses in chapter 16.2.25, *FLTK license*, page 352 and chapter 16.2.2, *GNU Lesser General Public License (LGPL)*, page 325 and

  Copyright (C) 1991-2012 The FLTK team

- VirtualBox contains portions of Expat which is governed by the license in chapter 16.2.26, *Expat license*, page 352 and

  Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper; Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

- VirtualBox contains portions of Fontconfig which is governed by the license in chapter 16.2.27, *Fontconfig license*, page 352 and

  Copyright (C) 2001, 2003 Keith Packard

- VirtualBox contains portions of Freetype which is governed by the license in chapter 16.2.28, *Freetype license*, page 353 and

  Copyright 2012 The FreeType Project (www.freetype.org). All rights reserved.

- VirtualBox may contain code from the WebM VP8 Codec SDK which is governed by the license in chapter 16.2.29, *VPX License*, page 355 and

  Copyright (c) 2010, The WebM Project authors. All rights reserved.

16.2 Licenses

16.2.1 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

16.2.2 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

16.2.3 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.

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 declaratory 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.]

16.2.4 MIT License

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.

16.2.5 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.

16.2.6 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
```

16.2.7 OpenSSL license

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscape's SSL.

This library is free for commercial and non-commercial use as long as the following conditions are adhered 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 routines 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 publicly 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.]

16.2.8 Slirp license

Copyright (c) 1995,1996 Danny Gasparovski. All rights reserved.

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, SPE-

CIAL, 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.

16.2.9 liblzf 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.

16.2.10 libpng license

The PNG Reference Library is supplied "AS IS". The Contributing Authors and Group 42, Inc. disclaim all warranties, expressed or implied, including, without limitation, the warranties of merchantability and of fitness for any purpose. The Contributing Authors and Group 42, Inc. assume no liability for direct, indirect, incidental, special, exemplary, or consequential damages, which may result from the use of the PNG Reference Library, even if advised of the possibility of such damage.

Permission is hereby granted to use, copy, modify, and distribute this source code, or portions hereof, for any purpose, without fee, subject to the following restrictions:

1. The origin of this source code must not be misrepresented.

2. Altered versions must be plainly marked as such and must not be misrepresented as being the original source.

3. This Copyright notice may not be removed or altered from any source or altered source distribution.

The Contributing Authors and Group 42, Inc. specifically permit, without fee, and encourage the use of this source code as a component to supporting the PNG file format in commercial products. If you use this source code in a product, acknowledgment is not required but would be appreciated.

16.2.11 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.

16.2.12 libxml license

Except where otherwise noted in the source code (e.g. the files hash.c, list.c and the trio files, which are covered by a similar licence but with different Copyright notices) all the files are:

Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved.

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 DANIEL VEILLARD 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.

Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

16.2.13 libxslt licenses

Licence for libxslt except libexslt:

Copyright (C) 2001-2002 Daniel Veillard. All Rights Reserved.

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

DANIEL VEILLARD 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.

Except as contained in this notice, the name of Daniel Veillard shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

Licence for libexslt:

Copyright (C) 2001-2002 Thomas Broyer, Charlie Bozeman and Daniel Veillard. All Rights Reserved.

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 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.

Except as contained in this notice, the name of the authors shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from him.

16.2.14 gSOAP Public License Version 1.3a

The gSOAP public license is derived from the Mozilla Public License (MPL1.1). The sections that were deleted from the original MPL1.1 text are 1.0.1, 2.1.(c),(d), 2.2.(c),(d), 8.2.(b), 10, and 11. Section 3.8 was added. The modified sections are 2.1.(b), 2.2.(b), 3.2 (simplified), 3.5 (deleted the last sentence), and 3.6 (simplified).

1 DEFINITIONS

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 now or hereafter owned or controlled by Initial Developer, to make, have made, use and sell ("offer to sell and import") the Original Code, Modifications, or portions thereof, but solely to the extent that any such patent is reasonably necessary to enable You to utilize, alone or in combination with other software, the Original Code, Modifications, or any combination or portions thereof.

(c)

(d)

2.2. Contributor Grant.

Subject to third party intellectual property claims, each Contributor hereby grants You a worldwide, 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 patents now or hereafter owned or controlled by Contributor, to make, have made, use and sell ("offer to sell and import") the Contributor Version (or portions thereof), but solely to the extent that any such patent is reasonably necessary to enable You to utilize, alone or in combination with other software, the Contributor Version (or portions thereof).

(c)

(d)

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 created by You will be provided to the Initial Developer in Source Code form and are subject to the terms of the License. 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.

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. 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. If you distribute executable versions containing Covered Code as part of a product, you must reproduce the notice in Exhibit B in the documentation and/or other materials provided with the product.

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.

3.8. Restrictions. You may not remove any product identification, copyright, proprietary notices or labels from gSOAP.

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.

Grantor 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.

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 phrase "gSOAP" 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 gSOAP 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, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, OF FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS, AND ANY WARRANTY THAT MAY ARISE BY REASON OF TRADE USAGE, CUSTOM, OR COURSE OF DEALING. WITHOUT LIMITING THE FOREGOING, YOU ACKNOWLEDGE THAT THE SOFTWARE IS PROVIDED "AS IS" AND THAT THE AUTHORS DO NOT WARRANT THE SOFTWARE WILL RUN UNINTERRUPTED OR ERROR FREE. LIMITED LIABILITY THE ENTIRE RISK AS TO RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. UNDER NO CIRCUMSTANCES WILL THE AUTHORS BE LIABLE FOR ANY SPECIAL, INDIRECT, INCIDENTAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES OF ANY KIND OR NATURE WHATSOEVER, WHETHER BASED ON CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), STRICT LIABILITY OR OTHERWISE, ARISING OUT OF OR IN ANY WAY RELATED TO THE SOFTWARE, EVEN IF THE AUTHORS HAVE BEEN ADVISED ON THE POSSIBILITY OF SUCH DAMAGE OR IF SUCH DAMAGE COULD HAVE BEEN REASONABLY FORESEEN, AND NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY EXCLUSIVE REMEDY PROVIDED. SUCH LIMITATION ON DAMAGES INCLUDES, BUT IS NOT LIMITED TO, DAMAGES FOR LOSS OF GOODWILL, LOST PROFITS, LOSS OF DATA OR SOFTWARE, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION OR IMPAIRMENT OF OTHER GOODS. IN NO EVENT WILL THE AUTHORS BE LIABLE FOR THE COSTS OF PROCUREMENT OF SUBSTITUTE SOFTWARE OR SERVICES. YOU ACKNOWLEDGE THAT THIS SOFTWARE IS NOT DESIGNED FOR USE IN ON-LINE EQUIP-

MENT IN HAZARDOUS ENVIRONMENTS SUCH AS OPERATION OF NUCLEAR FACILITIES, AIRCRAFT NAVIGATION OR CONTROL, OR LIFE-CRITICAL APPLICATIONS. THE AUTHORS EXPRESSLY DISCLAIM ANY LIABILITY RESULTING FROM USE OF THE SOFTWARE IN ANY SUCH ON-LINE EQUIPMENT IN HAZARDOUS ENVIRONMENTS AND ACCEPTS NO LIABILITY IN RESPECT OF ANY ACTIONS OR CLAIMS BASED ON THE USE OF THE SOFTWARE IN ANY SUCH ON-LINE EQUIPMENT IN HAZARDOUS ENVIRONMENTS BY YOU. FOR PURPOSES OF THIS PARAGRAPH, THE TERM "LIFE-CRITICAL APPLICATION" MEANS AN APPLICATION IN WHICH THE FUNCTIONING OR MALFUNCTIONING OF THE SOFTWARE MAY RESULT DIRECTLY OR INDIRECTLY IN PHYSICAL INJURY OR LOSS OF HUMAN LIFE. 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.

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.

11 MISCELLANEOUS.

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.

EXHIBIT A.

"The contents of this file are subject to the gSOAP Public License Version 1.3 (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.cs.fsu.edu/~engelen/soaplicense.html. 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 of the gSOAP Software is: stdsoap.h, stdsoap2.h, stdsoap.c, stdsoap2.c, stdsoap.cpp, stdsoap2.cpp, soapcpp2.h, soapcpp2.c, soapcpp2_lex.l, soapcpp2_yacc.y, error2.h, error2.c, symbol2.c, init2.c, soapdoc2.html, and soapdoc2.pdf, httpget.h, httpget.c, stl.h, stldeque.h, stllist.h, stlvector.h, stlset.h.

The Initial Developer of the Original Code is Robert A. van Engelen. Portions created by Robert A. van Engelen are Copyright (C) 2001-2004 Robert A. van Engelen, Genivia inc. All Rights Reserved.

Contributor(s): "_____." [Note: The text of this Exhibit A may differ slightly form 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.]

EXHIBIT B.

"Part of the software embedded in this product is gSOAP software. Portions created by gSOAP are Copyright (C) 2001-2004 Robert A. van Engelen, Genivia inc. All Rights Reserved. THE SOFTWARE IN THIS PRODUCT WAS IN PART PROVIDED BY GENIVIA INC 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."

16.2.15 Chromium licenses

16.2.15.1 Main license

Copyright (c) 2002, Stanford University All rights reserved.

Some portions of Chromium are copyrighted by individiual organizations. Please see the files COPYRIGHT.LLNL and COPYRIGHT.REDHAT for more information.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- Neither the name of Stanford University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 COPYRIGHT OWNER 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.

16.2.15.2 COPYRIGHT.LLNL file

This Chromium distribution contains information and code which is covered under the following notice:

Copyright (c) 2002, The Regents of the University of California. Produced at the Lawrence Livermore National Laboratory For details, contact: Randall Frank (rjfrank@llnl.gov). UCRL-CODE-2002-058 All rights reserved.

This file is part of Chromium. For details, see accompanying documentation.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the disclaimer below.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the disclaimer (as noted below) in the documentation and/or other materials provided with the distribution.

Neither the name of the UC/LLNL nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 REGENTS OF THE UNIVERSITY OF CALIFORNIA, THE U.S. DEPARTMENT OF ENERGY 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.

Additional BSD Notice

1. This notice is required to be provided under our contract with the U.S. Department of Energy (DOE). This work was produced at the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-ENG-48 with the DOE.

2. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights.

3. Also, reference herein to any specific commercial products, process, or services by trade name, trademark, manufacturer or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

16.2.15.3 COPYRIGHT.REDHAT file

This Chromium distribution contains information and code which is covered under the following notice:

Copyright 2001,2002 Red Hat Inc., Durham, North Carolina.

All Rights Reserved.

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 on 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 (including the next paragraph) 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 NON-INFRINGEMENT. IN NO EVENT SHALL RED HAT AND/OR THEIR SUPPLIERS 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.

16.2.16 curl license

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1996 - 2009, Daniel Stenberg, daniel@haxx.se.

All rights reserved.

Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

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 OF THIRD PARTY RIGHTS. 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.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

16.2.17 libgd license

Portions copyright 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002 by Cold Spring Harbor Laboratory. Funded under Grant P41-RR02188 by the National Institutes of Health.

Portions copyright 1996, 1997, 1998, 1999, 2000, 2001, 2002 by Boutell.Com, Inc.

Portions relating to GD2 format copyright 1999, 2000, 2001, 2002 Philip Warner.

Portions relating to PNG copyright 1999, 2000, 2001, 2002 Greg Roelofs.

Portions relating to gdttf.c copyright 1999, 2000, 2001, 2002 John Ellson (ellson@lucent.com).

Portions relating to gdft.c copyright 2001, 2002 John Ellson (ellson@lucent.com).

Portions copyright 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007 Pierre-Alain Joye (pierre@libgd.org).

Portions relating to JPEG and to color quantization copyright 2000, 2001, 2002, Doug Becker and copyright (C) 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, Thomas G. Lane. This software is based in part on the work of the Independent JPEG Group. See the file README-JPEG.TXT for more information.

Portions relating to WBMP copyright 2000, 2001, 2002 Maurice Szmurlo and Johan Van den Brande.

Permission has been granted to copy, distribute and modify gd in any context without fee, including a commercial application, provided that this notice is present in user-accessible supporting documentation.

This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for the authors of gd, not to interfere with your productive use of gd. If you have questions, ask. "Derived works" includes all programs that utilize the library. Credit must be given in user-accessible documentation.

This software is provided "AS IS." The copyright holders disclaim all warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to this code and accompanying documentation.

Although their code does not appear in gd, the authors wish to thank David Koblas, David Rowley, and Hutchison Avenue Software Corporation for their prior contributions.

16.2.18 BSD license from Intel

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- Neither the name of the Intel Corporation nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 COPYRIGHT OWNER 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.

16.2.19 libjpeg License

The authors make NO WARRANTY or representation, either express or implied, with respect to this software, its quality, accuracy, merchantability, or fitness for a particular purpose. This software is provided "AS IS", and you, its user, assume the entire risk as to its quality and accuracy.

This software is copyright (C) 1991-2010, Thomas G. Lane, Guido Vollbeding. All Rights Reserved except as specified below.

Permission is hereby granted to use, copy, modify, and distribute this software (or portions thereof) for any purpose, without fee, subject to these conditions:

(1) If any part of the source code for this software is distributed, then this README file must be included, with this copyright and no-warranty notice unaltered; and any additions, deletions, or changes to the original files must be clearly indicated in accompanying documentation.

(2) If only executable code is distributed, then the accompanying documentation must state that "this software is based in part on the work of the Independent JPEG Group".

(3) Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the authors accept NO LIABILITY for damages of any kind.

These conditions apply to any software derived from or based on the IJG code, not just to the unmodified library. If you use our work, you ought to acknowledge us.

Permission is NOT granted for the use of any IJG author's name or company name in advertising or publicity relating to this software or products derived from it. This software may be referred to only as "the Independent JPEG Group's software".

We specifically permit and encourage the use of this software as the basis of commercial products, provided that all warranty or liability claims are assumed by the product vendor.

ansi2knr.c is included in this distribution by permission of L. Peter Deutsch, sole proprietor of its copyright holder, Aladdin Enterprises of Menlo Park, CA. ansi2knr.c is NOT covered by the above copyright and conditions, but instead by the usual distribution terms of the Free Software Foundation; principally, that you must include source code if you redistribute it. (See the file ansi2knr.c for full details.) However, since ansi2knr.c is not needed as part of any program generated from the IJG code, this does not limit you more than the foregoing paragraphs do.

The Unix configuration script "configure" was produced with GNU Autoconf. It is copyright by the Free Software Foundation but is freely distributable. The same holds for its supporting scripts (config.guess, config.sub, ltmain.sh). Another support script, install-sh, is copyright by X Consortium but is also freely distributable.

The IJG distribution formerly included code to read and write GIF files. To avoid entanglement with the Unisys LZW patent, GIF reading support has been removed altogether, and the GIF writer has been simplified to produce "uncompressed GIFs". This technique does not use the LZW algorithm; the resulting GIF files are larger than usual, but are readable by all standard GIF decoders.

We are required to state that

"The Graphics Interchange Format(c) is the Copyright property of CompuServe Incorporated. GIF(sm) is a Service Mark property of CompuServe Incorporated."

16.2.20 x86 SIMD extension for IJG JPEG library license

Copyright 2009 Pierre Ossman <ossman@cendio.se> for Cendio AB

Copyright 2010 D. R. Commander

Based on

x86 SIMD extension for IJG JPEG library - version 1.02

Copyright (C) 1999-2006, MIYASAKA Masaru.

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.

16.2.21 FreeBSD license

The compilation of software known as FreeBSD is distributed under the following terms:

Copyright (c) 1992-2009 The FreeBSD Project. All rights reserved.

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.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS "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.

16.2.22 NetBSD license

Copyright (c) 1992, 1993 The Regents of the University of California. All rights reserved.

This software was developed by the Computer Systems Engineering group at Lawrence Berkeley Laboratory under DARPA contract BG 91-66 and contributed to Berkeley.

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. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "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 REGENTS 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.

16.2.23 PCRE license

PCRE is a library of functions to support regular expressions whose syntax and semantics are as close as possible to those of the Perl 5 language.

Release 8 of PCRE is distributed under the terms of the "BSD" licence, as specified below. The documentation for PCRE, supplied in the "doc" directory, is distributed under the same terms as the software itself.

The basic library functions are written in C and are freestanding. Also included in the distribution is a set of C++ wrapper functions, and a just-in-time compiler that can be used to optimize pattern matching. These are both optional features that can be omitted when the library is built.

THE BASIC LIBRARY FUNCTIONS. Written by: Philip Hazel; Email local part: ph10; Email domain: cam.ac.uk University of Cambridge Computing Service, Cambridge, England. Copyright (c) 1997-2012 University of Cambridge All rights reserved.

PCRE JUST-IN-TIME COMPILATION SUPPORT. Written by: Zoltan Herczeg; Email local part: hzmester; Emain domain: freemail.hu Copyright(c) 2010-2012 Zoltan Herczeg All rights reserved.

STACK-LESS JUST-IN-TIME COMPILER. Written by: Zoltan Herczeg; Email local part: hzmester; Emain domain: freemail.hu Copyright(c) 2009-2012 Zoltan Herczeg All rights reserved.

THE C++ WRAPPER FUNCTIONS. Contributed by: Google Inc. Copyright (c) 2007-2012, Google Inc. All rights reserved.

THE "BSD" LICENCE. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 COPYRIGHT OWNER 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.

16.2.24 libffi license

Copyright (c) 1996-2012 Anthony Green, Red Hat, Inc and others. See source files for details.

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.

16.2.25 FLTK license

December 11, 2001

The FLTK library and included programs are provided under the terms of the GNU Library General Public License (LGPL) with the following exceptions:

1. Modifications to the FLTK configure script, config header file, and makefiles by themselves to support a specific platform do not constitute a modified or derivative work.

   The authors do request that such modifications be contributed to the FLTK project - send all contributions through the "Software Trouble Report" on the following page:

   http://www.fltk.org/str.php

2. Widgets that are subclassed from FLTK widgets do not constitute a derivative work.

3. Static linking of applications and widgets to the FLTK library does not constitute a derivative work and does not require the author to provide source code for the application or widget, use the shared FLTK libraries, or link their applications or widgets against a user-supplied version of FLTK.

   If you link the application or widget to a modified version of FLTK, then the changes to FLTK must be provided under the terms of the LGPL in sections 1, 2, and 4.

4. You do not have to provide a copy of the FLTK license with programs that are linked to the FLTK library, nor do you have to identify the FLTK license in your program or documentation as required by section 6 of the LGPL.

   However, programs must still identify their use of FLTK. The following example statement can be included in user documentation to satisfy this requirement:

   [program/widget] is based in part on the work of the FLTK project (http://www.fltk.org).

16.2.26 Expat license

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd and Clark Cooper

Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006 Expat maintainers.

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.

16.2.27 Fontconfig license

Copyright (C) 2001, 2003 Keith Packard

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting

documentation, and that the name of the author(s) not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. The authors make no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

THE AUTHOR(S) DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL THE AUTHOR(S) BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

16.2.28 Freetype license

2006-Jan-27

Copyright 1996-2002, 2006 by David Turner, Robert Wilhelm, and Werner Lemberg

16.2.28.1 Introduction

The FreeType Project is distributed in several archive packages; some of them may contain, in addition to the FreeType font engine, various tools and contributions which rely on, or relate to, the FreeType Project.

This license applies to all files found in such packages, and which do not fall under their own explicit license. The license affects thus the FreeType font engine, the test programs, documentation and makefiles, at the very least.

This license was inspired by the BSD, Artistic, and IJG (Independent JPEG Group) licenses, which all encourage inclusion and use of free software in commercial and freeware products alike. As a consequence, its main points are that:

- We don't promise that this software works. However, we will be interested in any kind of bug reports. ('as is' distribution)
- You can use this software for whatever you want, in parts or full form, without having to pay us. ('royalty-free' usage)
- You may not pretend that you wrote this software. If you use it, or only parts of it, in a program, you must acknowledge somewhere in your documentation that you have used the FreeType code. ('credits')

We specifically permit and encourage the inclusion of this software, with or without modifications, in commercial products. We disclaim all warranties covering The FreeType Project and assume no liability related to The FreeType Project.

Finally, many people asked us for a preferred form for a credit/disclaimer to use in compliance with this license. We thus encourage you to use the following text:

Portions of this software are copyright (C) <year> The FreeType Project (www.freetype.org). All rights reserved.

Please replace <year> with the value from the FreeType version you actually use.

16.2.28.2 Legal Terms

0. Definitions Throughout this license, the terms 'package', 'FreeType Project', and 'FreeType archive' refer to the set of files originally distributed by the authors (David Turner, Robert Wilhelm, and Werner Lemberg) as the 'FreeType Project', be they named as alpha, beta or final release.

'You' refers to the licensee, or person using the project, where 'using' is a generic term including compiling the project's source code as well as linking it to form a 'program' or 'executable'. This program is referred to as 'a program using the FreeType engine'.

This license applies to all files distributed in the original FreeType Project, including all source code, binaries and documentation, unless otherwise stated in the file in its original, unmodified form as distributed in the original archive. If you are unsure whether or not a particular file is covered by this license, you must contact us to verify this.

The FreeType Project is copyright (C) 1996-2000 by David Turner, Robert Wilhelm, and Werner Lemberg. All rights reserved except as specified below.

1. No Warranty THE FREETYPE PROJECT IS PROVIDED 'AS IS' WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL ANY OF THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY DAMAGES CAUSED BY THE USE OR THE INABILITY TO USE, OF THE FREETYPE PROJECT.

2. Redistribution This license grants a worldwide, royalty-free, perpetual and irrevocable right and license to use, execute, perform, compile, display, copy, create derivative works of, distribute and sublicense the FreeType Project (in both source and object code forms) and derivative works thereof for any purpose; and to authorize others to exercise some or all of the rights granted herein, subject to the following conditions:

- Redistribution of source code must retain this license file ('FTL.TXT') unaltered; any additions, deletions or changes to the original files must be clearly indicated in accompanying documentation. The copyright notices of the unaltered, original files must be preserved in all copies of source files.
- Redistribution in binary form must provide a disclaimer that states that the software is based in part of the work of the FreeType Team, in the distribution documentation. We also encourage you to put an URL to the FreeType web page in your documentation, though this isn't mandatory.

These conditions apply to any software derived from or based on the FreeType Project, not just the unmodified files. If you use our work, you must acknowledge us. However, no fee need be paid to us.

3. Advertising Neither the FreeType authors and contributors nor you shall use the name of the other for commercial, advertising, or promotional purposes without specific prior written permission.

We suggest, but do not require, that you use one or more of the following phrases to refer to this software in your documentation or advertising materials: 'FreeType Project', 'FreeType Engine', 'FreeType library', or 'FreeType Distribution'.

As you have not signed this license, you are not required to accept it. However, as the FreeType Project is copyrighted material, only this license, or another one contracted with the authors, grants you the right to use, distribute, and modify it. Therefore, by using, distributing, or modifying the FreeType Project, you indicate that you understand and accept all the terms of this license.

4. Contacts There are two mailing lists related to FreeType:

- freetype@nongnu.org

  Discusses general use and applications of FreeType, as well as future and wanted additions to the library and distribution. If you are looking for support, start in this list if you haven't found anything to help you in the documentation.

- freetype-devel@nongnu.org

  Discusses bugs, as well as engine internals, design issues, specific licenses, porting, etc.

Our home page can be found at
http://www.freetype.org

16.2.29 VPX License

Copyright (c) 2010, The WebM Project authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 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.
- Neither the name of Google, nor the WebM Project, nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "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 COPYRIGHT HOLDER 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.

17 VirtualBox privacy information

Version 5, Dec 13, 2012

The Oracle Privacy Policies posted on http://www.oracle.com/html/privacy.html apply to your personal data collected and used by Oracle. The following privacy information describes in more detail which information is exchanged between the VirtualBox application and Oracle, and which information is collected by the virtualbox.org website.

§ 1 virtualbox.org. The "virtualbox.org" 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 (collectively, "anonymous data"). 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. The VirtualBox application may ask that the user optionally register with Oracle. If you choose to register, your name, e-mail address, country and company will be submitted to Oracle and stored together with the IP address of the submitter as well as product version and platform being used.

§ 4 Update notifications. The VirtualBox application may contact Oracle to find out whether a new version of VirtualBox has been released and notify the user if that is the case. In the process, anonymous data such as your IP address and a non-identifying counter, together with the product version and the platform being used, is sent so that the server can find out whether an update is available. By default, this check is performed once a day. You change this interval or disable these checks altogether in the VirtualBox preferences.

§ 5 Usage of personal information. Oracle may use anonymous and personal data collected by the means above 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. Oracle may also inform you about new product releases related to VirtualBox.

In no event will personal data without your express consent be provided to any third parties, unless Oracle may be required to do so by law or in connection with legal proceedings.

§ 6 Updates. Oracle may update the privacy policy at any time by posting a new version at http://www.oracle.com/html/privacy.html and the privacy information will be kept up to date in the documentation which comes with the VirtualBox application. You should check these places 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), SCSI, SAS, USB MSD, NVMe*, page 79.

AMD-V The hardware virtualization features built into modern AMD processors. See chapter 10.3, *Hardware vs. software virtualization*, page 239.

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 separate 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), SCSI, SAS, USB MSD, NVMe*, page 79.

B

BIOS Basic Input/Output System, the firmware built into most personal computers which is responsible of initializing the hardware after the computer has been turned on and then booting an operating system. VirtualBox ships with its own virtual BIOS that runs when a virtual machine is started.

C

COM Microsoft Component Object Model, a programming infrastructure for modular software. COM 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 92.

E

EFI Extensible Firmware Interface, a firmware built into computers which is designed to replace the aging BIOS. Originally designed by Intel, most modern operating systems can now boot on computers which have EFI instead of a BIOS built into them; see chapter 3.13, *Alternative firmware (EFI)*, page 59.

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), SCSI, SAS, USB MSD, NVMe*, page 79.

I/O APIC See APIC.

iSCSI Internet SCSI; see chapter 5.10, *iSCSI servers*, page 91.

M

MAC Media Access Control, a part of an Ethernet network card. A MAC address is a 6-byte number which identifies a network card. It is typically written in hexadecimal notation where the bytes are separated by colons, such as `00:17:3A:5E:CB:08`.

MSI Message Signaled Interrupts, as supported by modern chipsets such as the ICH9; see chapter 3.4.1, *"Motherboard" tab*, page 49. As opposed to traditional pin-based interrupts, with MSI, a small amount of data can accompany the actual interrupt message. This reduces the amount of hardware pins required, allows for more interrupts and better performance.

N

NAT Network Address Translation. A technique to share networking interfaces by which an interface modifies the source and/or target IP addresses of network packets according to specific rules. Commonly employed by routers and firewalls to shield an internal network

from the Internet, VirtualBox can use NAT to easily share a host's physical networking hardware with its virtual machines. See chapter 6.3, *Network Address Translation (NAT)*, page 94.

O

OVF Open Virtualization Format, a cross-platform industry standard to exchange virtual appliances between virtualization products; see chapter 1.14, *Importing and exporting virtual machines*, page 31.

P

PAE Physical Address Extension. This allows accessing more than 4 GB of RAM even in 32-bit environments; see chapter 3.3.2, *"Advanced" tab*, page 48.

PIC See APIC.

PXE Preboot Execution Environment, an industry standard for booting PC systems from remote network locations. It includes DHCP for IP configuration and TFTP for file transfer. Using UNDI, a hardware independent driver stack for accessing the network card from bootstrap code is available.

R

RDP Remote Desktop Protocol, a protocol developed by Microsoft as an extension to the ITU T.128 and T.124 video conferencing protocol. With RDP, a PC system can be controlled from a remote location using a network connection over which data is transferred in both directions. Typically graphics updates and audio are sent from the remote machine and keyboard and mouse input events are sent from the client. A VirtualBox extension package by Oracle provides VRDP, an enhanced implementation of the relevant standards which is largely compatible with Microsoft's RDP implementation. See chapter 7.1, *Remote display (VRDP support)*, page 104 for details.

S

SAS Serial Attached SCSI, an industry standard for hard disk interfaces. See chapter 5.1, *Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe*, page 79.

SATA Serial ATA, an industry standard for hard disk interfaces. See chapter 5.1, *Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe*, page 79.

SCSI Small Computer System Interface. An industry standard for data transfer between devices, especially for storage. See chapter 5.1, *Hard disk controllers: IDE, SATA (AHCI), SCSI, SAS, USB MSD, NVMe*, page 79.

SMP Symmetrical Multiprocessing, meaning that the resources of a computer are shared between several processors. These can either be several processor chips or, as is more common with modern hardware, multiple CPU cores in one processor.

SSD Solid-state drive, uses microchips for storing data in a computer system. Compared to classical hard-disks they are having no mechanical components like spinning disks.

T

TAR A widely used file format for archiving. Originally, this stood for "Tape ARchive" and was already supported by very early Unix versions for backing up data on tape. The file format is still widely used today, for example, with OVF archives (with an `.ova` file extension); see chapter 1.14, *Importing and exporting virtual machines*, page 31.

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 is 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 1.2, *Some terminology*, page 12 for details.

VMM Virtual Machine Manager – the component of VirtualBox that controls VM execution. See chapter 10.2, *VirtualBox executables and components*, page 237 for a list of VirtualBox components.

VRDE VirtualBox Remote Desktop Extension. This interface is built into VirtualBox to allow VirtualBox extension packages to supply remote access to virtual machines. A VirtualBox extension package by Oracle provides VRDP support; see chapter 7.1, *Remote display (VRDP support)*, page 104 for details.

VRDP See RDP.

VT-x The hardware virtualization features built into modern Intel processors. See chapter 10.3, *Hardware vs. software virtualization*, page 239.

X

xHCI eXtended Host Controller Interface, the interface that implements the USB 3.0 standard.

XML The eXtensible Markup Language, a metastandard for all kinds of textual information. XML only specifies how data in the document is organized generally and does not prescribe how to semantically organize content.

XPCOM Mozilla Cross Platform Component Object Model, a programming infrastructure developed by the Mozilla browser project which is similar to Microsoft COM and 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.