Chapter 9. Advanced Topics

Table of Contents

9.1. Automated Guest Logins

Oracle VM VirtualBox provides Guest Addition modules for Windows, Linux, and Oracle Solaris to enable automated logins on the guest.

When a guest operating system is running in a virtual machine, it might be desirable to perform coordinated and automated logins using credentials passed from the host. Credentials are user name, password, and domain name, where each value might be empty.

9.1.1. Automated Windows Guest Logins

Windows provides a modular system login subsystem, called Winlogon, which can be customized and extended by means of so-called GINA (Graphical Identification and Authentication) modules. In Windows Vista and later releases, the GINA modules were replaced with a new mechanism called credential providers. The Oracle VM VirtualBox Guest Additions for Windows come with both, a GINA and a credential provider module, and therefore enable any Windows guest to perform automated logins.

To activate the Oracle VM VirtualBox GINA or credential provider module, install the Guest Additions 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 Oracle VM VirtualBox GINA module, extract the Guest Additions as shown in Section 4.2.1.4, “Manual File Extraction”, and copy the `VBoxGINA.dll` file to the Windows `SYSTEM32` directory. In the registry, create the following key with a value of `VBoxGINA.dll`:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GinaDLL
```

> **Note**
>
> The Oracle VM VirtualBox GINA module is implemented as a wrapper around the `MSGINA.DLL` standard Windows GINA module. As a result, it might not work correctly with third-party GINA modules.

To manually install the Oracle VM VirtualBox credential provider module, extract the Guest Additions as shown in Section 4.2.1.4, "Manual File Extraction" and copy the `VBoxCredProv.dll` file to the Windows `SYSTEM32` directory. 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
```

All default values, the key named `Default`, must be set to `VBoxCredProv`.

Create the following string and assign it a value of `Apartment`.

```
HKEY_CLASSES_ROOT\CLSID\{275D3BCC-22BB-4948-A7F6-3A3054EBA92B}\InprocServer32\ThreadingModel
```

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 Oracle VM VirtualBox login modules, GINA or credential provider, using the Oracle VM VirtualBox Guest Additions device driver. When Windows is in *logged out* mode, the login modules will constantly poll for credentials and if they are present, a login will be attempted. After retrieving the credentials, the login modules will erase them so that the above command will have to be repeated for subsequent logins.

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. 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 Windows guest version, the following restrictions apply:

- For **Windows XP guests.** The login subsystem needs to be configured to use the classic login dialog, as the Oracle VM VirtualBox GINA module does not support the Windows XP-style welcome dialog.
- **Windows Vista, Windows 7, Windows 8, and Windows 10 guests.** The login 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 as Windows never renames user accounts internally.
- Automatic login handling of the built-in **Windows Remote Desktop Service**, formerly known as Terminal Services, is disabled by default. To enable it, create the following registry key with a `DWORD` value of `1`.

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\VirtualBox Guest Additions\AutoLogon
  ```

The following command forces Oracle VM 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.

9.1.2. Automated Linux and UNIX Guest Logins

Oracle VM VirtualBox provides a custom PAM module (Pluggable Authentication Module) which can be used to perform automated guest logins on platforms which support this framework. Virtually all modern Linux and UNIX distributions rely on PAM.

For automated logins on Ubuntu, or Ubuntu-derived, distributions using LightDM as the display manager. See Section 9.1.2.1, "Oracle VM VirtualBox Greeter for Ubuntu/LightDM".

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` module 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*`/other/` to the security modules directory. This is usually `/lib/security/` on 32-bit Linux guests or `/lib64/security/` on 64-bit Linux guests. 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 the GNOME Desktop Manager (GDM) to log in users automatically with the credentials passed by the host, configure the guest OS as follows:

1. Copy the `pam_vbox.so` module to the security modules directory. In this case, `/lib/security`.
2. Edit the PAM configuration file for GDM, found at `/etc/pam.d/gdm`. Add 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 Oracle VM 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, such as the ones provided by the Oracle VM 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** does not wait for credentials to arrive from the host. 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.

pam_vbox supports various guest property parameters that are located 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:

- `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 gets deleted no waiting for credentials will be performed and **pam_vbox** will act like before. This property must be set read-only for the guest (`RDONLYGUEST`).
- `CredsWaitAbort`: Aborts waiting for credentials when set to any value. Can be set from host and the guest.
- `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:

- `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`).
- `CredsMsgWaitTimeout`: Custom message showed when waiting for credentials by **pam_vbox** has timed out. For example, they 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 the correct flag set (`RDONLYGUEST`) the property is ignored and, depending on the property, a default value will be used. This can result in pam_vbox not waiting for credentials. Consult the appropriate syslog file for more information and use the `debug` option.

9.1.2.1. Oracle VM VirtualBox Greeter for Ubuntu/LightDM

Oracle VM VirtualBox comes with a greeter module, named **vbox-greeter**, that can be used with LightDM. LightDM is the default display manager for Ubuntu Linux and therefore can also be used for automated guest logins.

vbox-greeter does not need the **pam_vbox** module described in Section 9.1.2, "Automated Linux and UNIX Guest Logins"in order to function. It comes with its own authentication mechanism provided by LightDM. However, to provide maximum flexibility both modules can be used together on the same guest.

As with the **pam_vbox** module, **vbox-greeter** is shipped as part of the Guest Additions but it is not installed or activated on the guest OS by default. To install **vbox-greeter** automatically upon Guest Additions installation, use the `--with-autologon` option when starting the **VBoxLinuxAdditions.run** file:

```
# ./VBoxLinuxAdditions.run -- --with-autologon
```

For manual or postponed installation, copy the `vbox-greeter.desktop` file from `/opt/VBoxGuestAdditions-<version>/other/` to the `xgreeters` directory, which is usually `/usr/share/xgreeters/`. See your guest OS documentation for the name of the correct LightDM greeter directory.

The **vbox-greeter** module is installed by the Oracle VM VirtualBox Guest Additions installer and is located in `/usr/sbin/`. To enable **vbox-greeter** as the standard greeter module, edit the file `/etc/lightdm/lightdm.conf` as follows:

```
[SeatDefaults]
greeter-session=vbox-greeter
```

> **Note**
>
> - The LightDM server must be fully restarted in order for **vbox-greeter** to be used as the default greeter. As `root` on Ubuntu, run **service lightdm --full-restart** or restart the guest.
> - **vbox-greeter** is independent of the graphical session you choose, such as Gnome, KDE, or Unity. However, **vbox-greeter** does require FLTK 1.3 or later to implement 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 Section 9.1.2, "Automated Linux and UNIX Guest Logins".

In addition to the previously mentioned guest properties, **vbox-greeter** enables you to further customize its user interface. The following guest properties are located in the `/VirtualBox/GuestAdd/Greeter/` directory:

- `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`).
- `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`).
- `BannerPath`: Path to a `.PNG` file to use as a banner image on the top of the greeter. The image size must be 460 x 90 pixels, any bit depth. This property must be set read-only for the guest (`RDONLYGUEST`).
- `UseTheming`: Set to 1 for turning on the following theming options. This property must be set read-only for the guest (`RDONLYGUEST`).
- `Theme/BackgroundColor`: Hexadecimal RRGGBB color for the background. This property must be set read-only for the guest (`RDONLYGUEST`).
- `Theme/LogonDialog/HeaderColor`: Hexadecimal RRGGBB foreground color for the header text. This property must be set read-only for the guest (`RDONLYGUEST`).
- `Theme/LogonDialog/BackgroundColor`: Hexadecimal RRGGBB color for the login dialog background. This property must be set read-only for the guest (`RDONLYGUEST`).
- `Theme/LogonDialog/ButtonColor`: Hexadecimal RRGGBB background color for the login 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.

9.2. Advanced Configuration for Windows Guests

9.2.1. Automated Windows System Preparation

Microsoft offers a system preparation tool called Sysprep, to prepare a Windows system for deployment or redistribution. Some

Windows releases include Sysprep on the installation medium, but the tool is also available for download from the Microsoft web site. In a standard For most Windows versions, Sysprep is included in a default installation. Sysprep mainly consists of an executable called **sysprep.exe** which is invoked by the user to put the Windows installation into preparation mode.

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. See Section 4.9, "Guest Control of Applications" for details of how to use this feature with the special identifier `sysprep` as the program to execute, along with the user name `sysprep` and password `sysprep` for the credentials. Sysprep is then started with the required system rights.

> **Note**
>
> Specifying the location of **sysprep.exe** is **not possible**. Instead the following paths are used, based on the Windows release:
>
> - `C:\sysprep\sysprep.exe` for Windows XP and earlier
> - `%WINDIR%\System32\sysprep\sysprep.exe` for Windows Vista and later
>
> The Guest Additions will automatically use the appropriate path to execute the system preparation tool.

9.3. Advanced Configuration for Linux and Oracle Solaris Guests

9.3.1. Manual Setup of Selected Guest Services on Linux

The Oracle VM VirtualBox Guest Additions contain several different drivers. If you do not want to configure them all, use the following command to install the Guest Additions:

```
$ sh ./VBoxLinuxAdditions.run no_setup
```

After running this script, run the **rcvboxadd setup** command as `root` to compile the kernel modules.

On some 64-bit guests, you must replace `lib` with `lib64`. On older guests that do not run the **udev** service, you must add the **vboxadd** service to the default runlevel to ensure that the modules are loaded.

To set up the time synchronization service, add the **vboxadd-service** service to the default runlevel. To set up the X11 and OpenGL part of the Guest Additions, run the **rcvboxadd-x11 setup** command. Note that you do not need to enable additional services.

Use the **rcvboxadd setup** to recompile the guest kernel modules.

After compilation, reboot your guest to ensure that the new modules are loaded.

9.3.2. 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 Oracle VM VirtualBox Guest Additions includes drivers for X.Org. By default these drivers are in the following directory:

```
/opt/VBoxGuestAdditions-version/other/
```

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 Oracle VM VirtualBox. 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. In addition, for legacy X servers the correct `vboxmouse` driver must be loaded and associated with `/dev/mouse` or `/dev/psaux`. For most guests, a driver for a PS/2 mouse must be loaded and the correct vboxmouse driver must be associated with `/dev/vboxguest`.

The Oracle VM 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 Section 3.6, "Display Settings". The driver will offer a range of standard modes at least up to the default guest resolution for all active guest monitors. 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 legacy X Servers before version 1.3, you can also add your own modes to the X server configuration file. Add them to the "Modes" list in the "Display" subsection of the "Screen" section. For example, the following section 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
```

9.4. CPU Hot-Plugging

With virtual machines running modern server operating systems, Oracle VM VirtualBox supports CPU hot-plugging.

On a physical computer CPU hot-plugging would mean that a CPU can be added or removed while the machine is running. Oracle VM VirtualBox supports adding and removing of virtual CPUs while a virtual machine is running.

CPU hot-plugging works only with guest operating systems that support the feature. So far this applies only to Linux and Windows Server. Windows supports only hot-add, while Linux supports hot-add and hot-remove. To use this feature with more than 8 CPUs, a 64-bit Linux guest is required.

CPU hot-plugging is done using the **VBoxManage** command-line interface. First, hot-plugging needs to be enabled for a virtual machine:

```
$ VBoxManage modifyvm VM-name --cpu-hotplug on
```

The `--cpus` option is used to specify 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 **VBoxManage modifyvm --plug-cpu** and **VBoxManage modifyvm --unplug-cpu** commands, which take the number of the virtual CPU as a parameter, as follows:

```
$ VBoxManage modifyvm VM-name --plug-cpu 3
$ VBoxManage modifyvm VM-name --unplug-cpu 3
```

Note that CPU 0 can never be removed.

While the VM is running, CPUs can be added and removed with the **VBoxManage controlvm plugcpu** and **VBoxManage controlvm unplugcpu** commands instead, as follows:

```
$ VBoxManage controlvm VM-name plugcpu 3
$ VBoxManage controlvm VM-name unplugcpu 3
```

See Section 8.10, “VBoxManage modifyvm” and Section 8.20, “VBoxManage controlvm” 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
```

9.5. Webcam Passthrough

9.5.1. Using a Host Webcam in the Guest

Oracle VM VirtualBox includes a feature called *webcam passthrough*, which enables a guest to use a host webcam. This complements the general USB passthrough support which was the typical way of using host webcams in legacy releases. 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 Section 1.5, “Installing Oracle VM VirtualBox and Extension Packs”.

The host webcam can be attached to the VM using the **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 is as follows:

  ```
  alias "user friendly name"
  host path or identifier
  ```

  The alias can be used as a shortcut in other commands. Alias '.0' means the default video input device on the host. Alias '.1', '.2'means first, second video input device, and so on. The device order is host-specific.

- Attach a webcam to a running VM, as follows:

  ```
  VBoxManage controlvm VM name webcam attach [host_path|alias [settings]]
  ```

  This attaches a USB webcam device to the guest.

  The `settings` parameter is a string `Setting1=Value1;Setting2=Value2`, which enables you 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, as follows:

  ```
  VBoxManage controlvm VM-name webcam detach [host_path|alias]
  ```

- List the webcams attached to a running VM, as follows:

  ```
  VBoxManage controlvm VM-name webcam list
  ```

  The output contains the path or alias which was used in the **webcam attach** command for each attached webcam.

9.5.2. Windows Hosts

When the webcam device is detached from the host, the emulated webcam device is automatically detached from the guest.

9.5.3. macOS Hosts

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.

9.5.4. Linux and Oracle 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.

9.6. Advanced Display Configuration

9.6.1. Custom VESA Resolutions

Apart from the standard VESA resolutions, the Oracle VM VirtualBox VESA BIOS enables you to add up to 16 custom video modes which will be reported to the guest operating system. When using Windows guests with the Oracle VM 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 to be supplied to Linux:

```
vga = 0x200 | 0x160
vga = 864
```

For guest operating systems with Oracle VM VirtualBox Guest Additions, a custom video mode can be set using the video mode hint feature.

9.6.2. 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 Oracle VM 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, you can change it by issuing one of the following commands from the command line:

- Remove all limits on guest resolutions.

  ```
  VBoxManage setextradata global GUI/MaxGuestResolution any
  ```

- Manually specify a maximum resolution.

  ```
  VBoxManage setextradata global GUI/MaxGuestResolution widthxheight
  ```

- Restore the default settings to all guest VMs.

  ```
  VBoxManage setextradata global GUI/MaxGuestResolution auto
  ```

9.7. Advanced Storage Configuration

9.7.1. Using a Raw Host Hard Disk From a Guest

As an alternative to using virtual disk images as described in Chapter 5, *Virtual Storage*, Oracle VM VirtualBox can also present either entire physical hard disks or selected partitions as virtual disks to virtual machines.

With Oracle VM VirtualBox, this type of access is called *raw hard disk access*. It enables 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 compared to 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. For example, 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 Virtual Media Manager, see Section 5.3, "The Virtual Media Manager", or **VBoxManage** to assign the image to a virtual machine.

9.7.1.1. 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.

On a Linux host, 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, use the following command:

```
$ VBoxManage createmedium disk --filename path-to-file.vmdk --format=VMDK
 --variant RawDisk --property RawDrive=/dev/sda
```

This creates the *path-to-file*.vmdk file image that must be an absolute path. All data is read and written from /dev/sda.

On a Windows host, instead of the above device specification, for example use \\.\PhysicalDrive0. On a macOS host, instead of the above device specification use for example /dev/rdisk1. Note that on Mac 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, such as Windows, 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 as follows:

```
$ 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.

9.7.1.2. 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. This means that you can 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, on a Linux host, use the command:

```
$ VBoxManage createmedium disk --filename path-to-file.vmdk --format=VMDK
--variant RawDisk --property RawDrive=/dev/sda --property Partitions=1,5
```

The command is identical to the one for full hard disk access, except for the additional --property Partitions=1,5 parameter. This example would create the image *path-to-file*.vmdk, which must be absolute, and partitions 1 and 5 of /dev/sda would be made accessible to the guest.

Oracle VM 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 for example \\.\PhysicalDrive0. On a macOS host, instead of the above device specification use /dev/rdisk1, for example. Note that on OS X you can only use partitions which are not mounted. Unmount the respective disk first using *diskutil unmountDisk /dev/diskX*. Partition numbers are the same on Linux, Windows, and macOS hosts.

The numbers for the list of partitions can be taken from the output of the following command:

```
$ VBoxManage list hostdrives
```

The output lists available drives and their partitions with 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 createmedium disk --filename path-to-file.vmdk --format=VMDK
--variant RawDisk --property RawDrive=/dev/sda --property Partitions=1,5
--property Relative=1
```

When used from a virtual machine, the image will then refer not to the entire disk, but only to the individual partitions. In this 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. For example, to replace the Linux boot loader that is used on the host by another boot loader. This enables for example the guest to boot directly to Windows, while the host boots Linux from the "same" disk. For this purpose the `--property-file BootSector=`*`path-to-file-with-boot-sector`* 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 createmedium disk --filename path-to-file.vmdk --format=VMDK
--variant RawDisk --property RawDrive=/dev/sda --property Partitions=1,5
--property-file BootSector=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.

9.7.2. Configuring the Hard Disk Vendor Product Data (VPD)

Oracle VM 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, you can mark the drive as having a non-rotational medium by using the following command:

```
$ 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.

9.7.3. Access iSCSI Targets Using Internal Networking

As an experimental feature, Oracle VM VirtualBox enables access to an iSCSI target running in a virtual machine which is configured to use Internal Networking mode. See Section 5.10, “iSCSI Servers”, Section 6.6, “Internal Networking”, and Section 8.26, “VBoxManage storageattach”.

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, as follows:

```
$ VBoxManage storageattach ... --medium iscsi --server 10.0.9.30 \
--target iqn.2008-12.com.sun:sampletarget --intnet
```

Compared to a regular iSCSI setup, the 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.

9.8. Fine Tuning the Oracle VM VirtualBox NAT Engine

9.8.1. 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 the NAT network needs to be changed, use 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`.

9.8.2. Configuring the Boot Server (Next Server) of a NAT Network Interface

For network booting in NAT mode, by default Oracle VM 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
```

9.8.3. Tuning TCP/IP Buffers for NAT

The Oracle VM 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, where 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.

9.8.4. Binding NAT Sockets to a Specific Interface

By default, Oracle VM 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 you want to change this behavior, you

can tell the NAT engine to bind to a particular IP address instead. For example, 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. Ensure that this interface is up and running before changing the NAT bind address.

9.8.5. 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
```

9.8.6. 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 Oracle VM 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.

9.8.6.1. 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 enables 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" hostname
```

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 name pattern can include the following wildcard characters: pipe (|), question mark (?), and asterisk (*).

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 do not have any effect.

9.8.7. 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. You can change the NAT mode by using the following commands:

```
$ VBoxManage modifyvm VM-name \
--nataliasmode1 proxyonly

$ 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.

9.9. Configuring the BIOS DMI Information

The DMI data that Oracle VM 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 Oracle VM 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 example:

```
$ 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 the following command:

```
$ dmidecode -t0
```

The DMI system information can be obtained as follows:

```
$ dmidecode -t1
```

9.10. Configuring Custom ACPI Tables

You can configure Oracle VM VirtualBox to present up to four custom ACPI tables to the guest. Use a command such as the following to configure custom ACPI tables. Note that `CustomTable1`, `CustomTable2`, and `CustomTable3` are available in addition to `CustomTable0`.

```
$ VBoxManage setextradata VM-name \
"VBoxInternal/Devices/acpi/0/Config/CustomTable0" "/path-to-table.bin"
```

Configuring custom ACPI tables can for example avoid the need for asking for a new product key on Windows Vista, Windows 7, Windows 8 and later guests. On Linux hosts, one of the system's ACPI tables can be read from `/sys/firmware/acpi/tables/`.

9.11. Fine Tuning Timers and Time Synchronization

9.11.1. Configuring the Guest Time Stamp Counter (TSC) to Reflect Guest Execution

By default, Oracle VM 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 time stamp counter (TSC) 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.

9.11.2. 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 Oracle VM 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 Section 9.11.3, “Tuning the Guest Additions Time Synchronization Parameters”.

9.11.3. Tuning the Guest Additions Time Synchronization Parameters

The Oracle VM 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/property" value
```

property 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, which means as follows:

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. Do not 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. This is the 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.

9.11.4. Disabling the Guest Additions Time Synchronization

Once installed and started, the Oracle VM 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
```

9.12. Installing the Alternate Bridged Networking Driver on Oracle Solaris 11 Hosts

Oracle VM VirtualBox includes a network filter driver that utilizes Oracle Solaris 11's Crossbow functionality. By default, this new driver

is installed for Oracle Solaris 11 hosts that have support for it.

To force installation of the older STREAMS based network filter driver, execute as root the following command before installing the Oracle VM 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 Oracle VM VirtualBox package:

```
$ touch /etc/vboxinst_vboxbow
```

To check which driver is currently being used by Oracle VM VirtualBox, execute:

```
$ modinfo | grep vbox
```

If the output contains "vboxbow", it indicates Oracle VM VirtualBox is using the Crossbow network filter driver, while the name "vboxflt" indicates usage of the older STREAMS network filter.

9.13. Oracle VM VirtualBox VNIC Templates for VLANs on Oracle Solaris 11 Hosts

Oracle VM VirtualBox supports Virtual Network Interface (VNIC) templates for configuring VMs over VLANs. An Oracle VM VirtualBox VNIC template is a VNIC whose name starts with `vboxvnic_template`. The string is case-sensitive.

On Oracle Solaris 11 hosts, when Crossbow-based bridged networking is used, a VNIC template may be used to specify the VLAN ID to use while bridging over a network link.

The following is an example of how to use a VNIC template to configure a VM over a VLAN. Create an Oracle VM 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 the following command:

```
$ 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 the **dladm** command.

VNIC templates can be created with additional properties such as bandwidth limits and CPU fanout. Refer to your Oracle Solaris network documentation for details. The additional properties are also applied to VMs which bridge using the VNIC template.

9.14. Configuring Multiple Host-Only Network Interfaces on Oracle Solaris Hosts

By default Oracle VM VirtualBox provides you with one host-only network interface. Adding more host-only network interfaces on Oracle Solaris hosts requires manual configuration. Here is 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:

```
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 Oracle 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 vboxnet*x* 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 settings for the newly added interfaces 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 Oracle VM VirtualBox installer only updates these configuration files for the one "vboxnet0" interface it creates by default.

9.15. Configuring the Oracle VM VirtualBox CoreDumper on Oracle Solaris Hosts

Oracle VM VirtualBox is capable of producing its own core files for extensive debugging when things go wrong. Currently this is only available on Oracle Solaris hosts.

The Oracle VM 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, as follows:

```
$ 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 Oracle VM VirtualBox process has sufficient permissions to write files to this directory. If you skip this command and do not specify any core dump directory, the current directory of the Oracle VM VirtualBox executable will be used. This 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 Oracle VM 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 Oracle VM 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 Oracle VM 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 the following command:

```
$ kill -s SIGUSR2 VM-process-id
```

The Oracle VM VirtualBox CoreDumper creates core files of the form `core.vb.`*`process-name`*`.`*`process-ID`* such as `core.vb.VBoxHeadless.11321`.

9.16. Oracle VM VirtualBox and Oracle Solaris Kernel Zones

Oracle Solaris kernel zones on x86-based systems make use of hardware-assisted virtualization features like Oracle VM VirtualBox does. However, for kernel zones and Oracle VM VirtualBox to share this hardware resource, they need to cooperate.

By default, due to performance reasons, Oracle VM VirtualBox acquires the hardware-assisted virtualization resource (VT-x/AMD-V)

globally on the host machine and uses it until the last Oracle VM VirtualBox VM that requires it is powered off. This prevents other software from using VT-x/AMD-V during the time Oracle VM VirtualBox has taken control of it.

Oracle VM 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 Oracle VM 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 Oracle VM VirtualBox settings which will continue to persist across host-reboots and Oracle VM VirtualBox upgrades.

9.17. Locking Down VirtualBox Manager

9.17.1. Customizing VirtualBox Manager

There are several advanced customization settings for locking down VirtualBox Manager. Locking down means removing some features that the user should not see.

```
VBoxManage setextradata global GUI/Customizations property[,property ...]
```

property is one of the following properties:

`noSelector`

Do not allow users to start 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 VirtualBox Manager customizations use the following command:

```
$ VBoxManage setextradata global GUI/Customizations
```

9.17.2. 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 property true
```

property can be any of the following:

`GUI/HideDetails`

Do not show the VM configuration of a certain VM. The details window will remain just empty if this VM is selected.

`GUI/PreventReconfiguration`

Do not 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.

Note that these settings do not prevent the user from reconfiguring the VM by using the **VBoxManage modifyvm** command.

9.17.3. Configure VM Selector Menu Entries

You can disable certain entries in the global settings page of the VM selector:

```
$ VBoxManage setextradata global GUI/RestrictedGlobalSettingsPages property[,property...]
```

property is one of the following:

`General`

Do not show the **General** settings pane.

`Input`

Do not show the **Input** settings pane.

`Update`

Do not show the **Update** settings pane.

`Language`

Do not show the **Language** settings pane.

`Display`

Do not show the **Display** settings pane.

`Network`

Do not show the **Network** settings pane.

`Extensions`

Do not show the **Extensions** settings pane.

`Proxy`

Do not show the **Proxy** settings pane.

This is a global setting. You can specify any combination of properties. To restore the default behavior, use the following command:

```
$ VBoxManage setextradata global GUI/RestrictedGlobalSettingsPages
```

9.17.4. Configure VM Window Menu Entries

You can disable certain menu actions in the VM window:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeMenus OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All`

Do not show any menu in the VM window.

`Application`

Do not show **Application/File** menu in the VM window.

`Machine`

Do not show the **Machine** menu in the VM window.

`View`

Do not show the **View** menu in the VM window.

`Input`

Do not show **Input** menu in the VM window.

`Devices`

Do not show the **Devices** menu in the VM window.

`Help`

Do not show the **Help** menu in the VM window.

`Debug`

Do not 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 or global setting. Any combination of the above is allowed. To restore the default behavior, use the following command:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeMenus
```

You can also disable certain menu actions of certain menus. Use the following command to disable certain actions of the **Application** menu. This is only available on macOS hosts.

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeApplicationMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All`

Do not show any menu item in this menu.

`About`

Do not show the **About** menu item in this menu.

`Preferences`

Do not show the **Preferences** menu item in this menu.

`NetworkAccessManager`

Do not show the **Network Operations Manager** menu item in this menu.

`ResetWarnings`

Do not show the **Reset All Warnings** menu item in this menu.

`Close`

Do not show the **Close** menu item in this menu.

This is a per-VM or global setting. Any combination of the above is allowed. To restore the default behavior, use the following command:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeMenus
```

Use the following command to disable certain actions of the **Machine** menu:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeMachineMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All`

Do not show any menu item in this menu.

`SettingsDialog`

Do not show the **Settings** menu item in this menu.

`TakeSnapshot`

Do not show the **Take Snapshot...** menu item in this menu.

`InformationDialog`

Do not show the **Session Information...** menu item in this menu.

`FileManagerDialog`

Do not show the **File Manager...** menu item in this menu.

`Pause`

Do not show the **Pause** menu item in this menu.

`Reset`

Do not show the **Reset** menu item in this menu.

`Shutdown`

Do not show the **ACPI Shutdown** menu item in this menu.

This is a per-VM or global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeMachineMenuActions
```

Use the following command to disable certain actions of the **View** menu:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeViewMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All`

Do not show any menu item in this menu.

`Fullscreen`

Do not show the **Full-screen Mode** menu item in this menu.

`Seamless`

Do not show the **Seamless Mode** menu item in this menu.

`Scale`

Do not show the **Scaled Mode** menu item in this menu.

`GuestAutoresize`

Do not show the **Auto-resize Guest Display** menu item in this menu.

`AdjustWindow`

Do not show the **Adjust Window Size** menu item in this menu.

`TakeScreenshot`

Do not show the **Take Screenshot...** menu item in this menu.

`Recording`

Do not show the **Recording** menu item in this menu.

`VRDEServer`

Do not show the **Remote Display** menu item in this menu.

`MenuBar`

Do not show the **Menu Bar** menu item in this menu.

`MenuBarSettings`

Do not show the **Menu Bar Settings...** menu item in this menu.

`StatusBar`

Do not show the **Status Bar** menu item in this menu.

`StatusbarSettings`

Do not show the **Statusbar Settings...** menu item in this menu.

This is a per-VM or global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeViewMenuActions
```

Use the following command to disable certain actions of the **Input** menu:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeInputMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`All`

Do not show any menu item in this menu.

`Keyboard`

Do not show the **Keyboard** menu item in this menu.

`KeyboardSettings`

Do not show the **Keyboard Settings...** menu item in this menu.

`SoftKeyboard`

Do not show the **Soft Keyboard...** menu item in this menu.

`TypeCAD`

Do not show the **Insert Ctrl-Alt-Del** menu item in this menu.

`TypeCABS`

Do not show the **Insert Ctrl-Alt-Backspace** menu item in this menu.

`TypeCtrlBreak`

Do not show the **Insert Ctrl-Break** menu item in this menu.

`TypeInsert`

Do not show the **Insert Insert** menu item in this menu.

`TypePrintScreen`

Do not show the **Insert Print Screen** menu item in this menu.

`TypeAltPrintScreen`

Do not show the **Insert Alt Print Screen** menu item in this menu.

`TypeHostKeyCombo`

Do not show the **Insert Host Key Combo** menu item in this menu.

`MouseIntegration`

Do not show the **MouseIntegration** menu item in this menu.

This is a per-VM or global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeInputMenuActions
```

Use the following command to disable certain actions of the **Devices** menu:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeDevicesMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords to disable actions in the **Devices** menu:

`All`

Do not show any menu item in this menu.

`HardDrives`

Do not show the **Hard Disks** menu item in this menu.

`OpticalDevices`

Do not show the **Optical Devices** menu item in this menu.

`FloppyDevices`

Do not show the **Floppy Drives** menu item in this menu.

`Audio`

Do not show the **Audio** menu item in this menu.

`Network`

Do not show the **Network** menu item in this menu.

`NetworkSettings`

Do not show the **Network Settings** menu item in this menu.

`USBDevices`

Do not show the **USB** menu item in this menu.

`WebCams`

Do not show the **WebCams** menu item in this menu.

`SharedFolders`

Do not show the **Shared Folders** menu item in this menu.

`SharedFoldersSettings`

Do not show the **Shared Folders Settings...** menu item in this menu.

`SharedClipboard`

Do not show the **Shared Clipboard** menu item in this menu.

`DragAndDrop`

Do not show the **Drag and Drop** menu item in this menu.

`InstallGuestTools`

Do not show the **Insert Guest Additions CD image...** menu item in this menu.

This is a per-VM or global or global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeDevicesMenuActions
```

Use the following command to disable certain actions of the **Debug** menu:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeDebuggerMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords to disable actions in the *Debug* menu, which is normally completely disabled:

`All`

Do not show any menu item in this menu.

`Statistics`

Do not show the **Statistics...** menu item in this menu.

`CommandLine`

Do not show the **Command Line...** menu item in this menu.

`Logging`

Do not show the **Logging...** menu item in this menu.

`LogDialog`

Do not show the **Show Log...** menu item in this menu.

`GuestControlConsole`

Do not show the **Guest Control Terminal...** menu item in this menu.

This is a per-VM or global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeDebuggerMenuActions
```

Use the following command to disable certain actions of the **View** menu:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeHelpMenuActions OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords to disable actions in the **Help** menu, which is normally completely disabled:

`All`

Do not show any menu item in this menu.

`Contents`

Do not show the **Contents...** menu item in this menu.

`WebSite`

Do not show the **VirtualBox Web Site...** menu item in this menu.

`BugTracker`

Do not show the **VirtualBox Bug Tracker...** menu item in this menu.

`Forums`

Do not show the **VirtualBox Forums...** menu item in this menu.

`Oracle`

Do not show the **Oracle Web Site...** menu item in this menu.

`About`

Do not show the **About VirtualBox...** menu item in this menu. Only for non-macOS hosts.

This is a per-VM or global setting. Any combination of the above is allowed. To restore the default behavior, use

```
VBoxManage setextradata "VM name"|global GUI/RestrictedRuntimeHelpMenuActions
```

9.17.5. Configure VM Window Status Bar Entries

You can disable certain status bar items:

```
VBoxManage setextradata "VM name"|global GUI/RestrictedStatusBarIndicators OPTION[,OPTION...]
```

where `OPTION` is one of the following keywords:

`HardDisks`

Do not 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`

Do not 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`

Do not 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`

Do not 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`

Do not show the USB icon in the status bar.

`SharedFolders`

Do not show the shared folders icon in the status bar.

`Capture`

Do not show the capture icon in the status bar.

`Features`

Do not show the CPU features icon in the status bar.

`Mouse`

Do not show the mouse icon in the status bar.

`Keyboard`

Do not show the keyboard icon in the status bar.

This is a per-VM or global 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"|global GUI/RestrictedStatusBarIndicators
```

9.17.6. Configure VM Window Visual Modes

You can disable certain VM visual modes:

```
$ VBoxManage setextradata VM-name GUI/RestrictedVisualStates property[,property...]
```

property is one of the following:

`Fullscreen`

Do not allow to switch the VM into full screen mode.

`Seamless`

Do not allow to switch the VM into seamless mode.

`Scale`

Do not allow to switch the VM into scale mode.

This is a per-VM setting. You can specify any combination of properties. To restore the default behavior, use the following command:

```
$ VBoxManage setextradata VM-name GUI/RestrictedVisualStates
```

9.17.7. 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 Oracle VM 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 table shows the possible Host key actions, together with their default Host key shortcut. Setting an action to None will disable that Host key action.

Table 9.1. Host Key Customization

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 off the VM without saving the state
Close	Q	Show the Close VM dialog
FullscreenMode	F	Switch the VM into full screen mode
SeamlessMode	L	Switch the VM into seamless mode
ScaleMode	C	Switch the VM into scaled mode
GuestAutoResize	G	Automatically resize the guest window
WindowAdjust	A	Immediately resize the guest window
PopupMenu	Home	Show the popup menu in full screen mode and seamless mode
SettingsDialog	S	Open the VM Settings dialog
InformationDialog	N	Show the VM Session 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 full screen mode and seamless mode, use the following command:

```
$ VBoxManage setextradata global GUI/Input/MachineShortcuts "FullscreenMode=None,SeamlessMode=None"
```

9.17.8. Action when Terminating the VM

You can disallow certain actions when terminating a VM. To disallow specific actions, use the following command:

```
$ VBoxManage setextradata VM-name GUI/RestrictedCloseActions property[,property...]
```

property is one of the following:

`SaveState`

Do not allow the user to save the VM state when terminating the VM.

`Shutdown`

Do not allow the user to shutdown the VM by sending the ACPI power-off event to the guest.

`PowerOff`

Do not allow the user to power off the VM.

`PowerOffRestoringSnapshot`

Do not allow the user to return to the last snapshot when powering off the VM.

`Detach`

Do not allow the user to detach from the VM process if the VM was started in separate mode.

This is a per-VM setting. You can specify any combination of properties. If all properties are specified, the VM cannot be shut down.

9.17.9. 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. Use the following command:

```
$ VBoxManage setextradata VM-name GUI/DefaultCloseAction action
```

action is one of the following:

`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. You can specify any combination of properties. If all properties are specified, the VM cannot be shut down.

9.17.10. 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 as follows:

```
$ VBoxManage setextradata VM-name GUI/GuruMeditationHandler mode
```

mode is one of the following:

`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.

9.17.11. 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 has not yet started the Oracle VM VirtualBox Guest Additions. For instance, the guest is booting or the Guest Additions are not installed, or if the pointing device is configured as a 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 as follows:

```
VBoxManage setextradata VM-name GUI/MouseCapturePolicy mode
```

mode is one of the following:

`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.

9.17.12. Requesting Legacy Full-Screen Mode

Oracle VM 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. Oracle VM VirtualBox can be configured to use a legacy method of switching to full-screen mode instead, by using the command:

```
VBoxManage setextradata global GUI/Fullscreen/LegacyMode true
```

You can go back to the default method by using the following command:

```
VBoxManage setextradata global GUI/Fullscreen/LegacyMode
```

This is a global setting.

9.17.13. Removing Certain Modes of Networking From the GUI

It is possible to remove networking modes from Oracle VM VirtualBox GUI. To do this, use the following command:

```
VBoxManage setextradata global GUI/RestrictedNetworkAttachmentTypes property[,property...]
```

property is one of the following:

`NAT`

Remove the **NAT** option from the GUI.

`NATNetwork`

Remove the **NAT network** option from the GUI.

`BridgedAdapter`

Remove the **Bridged networking** option from the GUI.

`InternalNetwork`

Remove the **Internal networking** option from the GUI.

`HostOnlyAdapter`

Remove the **Host Only networking** option from the GUI.

`GenericDriver`

Remove the **Generic networking** option from the GUI.

This is a global setting. You can specify any combination of properties. To restore the default behavior, use the following command:

```
VBoxManage setextradata global GUI/RestrictedNetworkAttachmentTypes
```

9.18. Starting the Oracle VM VirtualBox Web Service Automatically

The Oracle VM VirtualBox web service, **vboxwebsrv**, is used for controlling Oracle VM VirtualBox remotely. It is documented in detail in the Oracle VM VirtualBox Software Development Kit (SDK). See Chapter 11, *Oracle VM VirtualBox Programming Interfaces*. Web service start scripts are available for supported host operating systems. The following sections describe how to use the scripts. The Oracle VM VirtualBox web service is never started automatically as a result of a standard installation.

9.18.1. Linux: Starting the Web Service With 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 following table all start with the `VBOXWEB_` prefix string. For example: `VBOXWEB_HOST` and `VBOXWEB_PORT`.

Table 9.2. Web Service Configuration Parameters

Parameter	Description	Default
`USER`	The user which the web service runs as	
`HOST`	The host to bind the web service to	localhost
`PORT`	The port to bind the web service to	18083
`SSL_KEYFILE`	Server key and certificate file, in PEM format	
`SSL_PASSWORDFILE`	File name for password to server key	
`SSL_CACERT`	CA certificate file, in 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 log file size to trigger rotation, in bytes	1MB
`LOGINTERVAL`	Maximum time interval to trigger log rotation, in seconds	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.

9.18.2. Oracle Solaris: Starting the Web Service With SMF

On Oracle Solaris hosts, the Oracle VM VirtualBox web service daemon is integrated into the SMF framework. You can change the parameters, but do not 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 Section 9.18.1, “Linux: Starting the Web Service With init” showing the parameter names and defaults also applies for Oracle Solaris. The parameter names must be changed to lowercase and a prefix of `config/` has to be added. For example: `config/user` or `config/ssl_keyfile`. If you make any change, do not 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 are used when enabling the service. Check the current property settings as follows:

```
svcprop -p config svc:/application/virtualbox/webservice:default
```

When everything is configured correctly you can start the Oracle VM VirtualBox web service with the following command:

```
svcadm enable svc:/application/virtualbox/webservice:default
```

For more information about SMF, please refer to the Oracle Solaris documentation.

9.18.3. macOS: Starting the Web Service With launchd

On macOS, launchd is used to start the Oracle VM 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.

9.19. Oracle VM VirtualBox Watchdog

The memory ballooning service, formerly known as **VBoxBalloonCtrl**, was renamed to VBoxWatchdog. This service now incorporates the following host services that are meant to be run in a server environment:

- **Memory ballooning control.** This service automatically takes care of a VM's configured memory balloon. See Section 4.10.1, "Memory Ballooning". This service 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.** This service provides a way to detect whether the host cannot reach the specific Oracle VM 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 using the 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.

9.19.1. 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 using the command line, as follows:

```
--balloon-max <Size in MB>
```

Using a per-VM basis extradata value, as follows:

```
VBoxManage setextradata <VM-Name> VBoxInternal2/Watchdog/BalloonCtrl/BalloonSizeMax <Size in MB>
```

Using a global extradata value, as follows:

```
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 using command line, as follows:

```
--balloon-inc <Size in MB>
```

Using a global extradata value, as follows:

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonIncrementMB <Size in MB>
```

The default ballooning increment is 256 MB if not specified.

The same options apply for a ballooning decrement. Using the command line, as follows:

```
--balloon-dec <Size in MB>
```

Using a global extradata value, as follows:

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonDecrementMB <Size in MB>
```

The default ballooning decrement is 128 MB if not specified.

The lower limit in MB for a balloon can be defined using the command line, as follows:

```
--balloon-lower-limit <Size in MB>
```

Using a global extradata value, as follows:

```
VBoxManage setextradata global VBoxInternal2/Watchdog/BalloonCtrl/BalloonLowerLimitMB <Size in MB>
```

The default lower limit is 128 MB if not specified.

9.19.2. Host Isolation Detection

To detect whether a host is being isolated, that is, the host cannot reach the Oracle VM 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.

Set the groups handled by the host isolation detection using the following command line:

```
--apimon-groups=<string[,stringN]>
```

Using a global extradata value, as follows:

```
VBoxManage setextradata global VBoxInternal2/Watchdog/APIMonitor/Groups <string[,stringN]>
```

Set the host isolation timeout using the following command line:

```
--apimon-isln-timeout=<ms>
```

Using a global extradata value, as follows:

```
VBoxManage setextradata global VBoxInternal2/Watchdog/APIMonitor/IsolationTimeoutMS <ms>
```

Set the actual host isolation response using the following command line:

```
--apimon-isln-response=<cmd>
```

Using a global extradata value, as follows:

```
VBoxManage setextradata global VBoxInternal2/Watchdog/APIMonitor/IsolationResponse <cmd>
```

The following response commands are available:

- `none`. This has no effect.
- `pause`. Pauses the execution of a VM.
- `poweroff`. 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`. Saves the current machine state and powers off the VM afterwards. If saving the machine state fails the VM will be paused.
- `shutdown`. 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.

9.19.3. More Information

For more advanced options and parameters like verbose logging check the built-in command line help accessible with `--help`.

9.19.4. Linux: Starting the Watchdog Service With 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 following table all start with the `VBOXWATCHDOG_` prefix string. For example: `VBOXWATCHDOG_BALLOON_INTERVAL` and `VBOXWATCHDOG_LOGSIZE`. Legacy parameters such as `VBOXBALLOONCTRL_INTERVAL` can still be used.

Table 9.3. Oracle VM VirtualBox Watchdog Configuration Parameters

Parameter	Description	Default
`USER`	The user which the watchdog service runs as	
`ROTATE`	Number of log files, 0 disables log rotation	10
`LOGSIZE`	Maximum log file size to trigger rotation, in bytes	1MB
`LOGINTERVAL`	Maximum time interval to trigger log rotation, in seconds	1 day
`BALLOON_INTERVAL`	Interval for checking the balloon size, in milliseconds	30000
`BALLOON_INCREMENT`	Balloon size increment, in megabytes	256
`BALLOON_DECREMENT`	Balloon size decrement, in megabytes	128
`BALLOON_LOWERLIMIT`	Balloon size lower limit, in megabytes	64
`BALLOON_SAFETYMARGIN`	Free memory required for decreasing the balloon size, in megabytes	1024

9.19.5. Oracle Solaris: Starting the Watchdog Service With SMF

On Oracle Solaris hosts, the Oracle VM VirtualBox watchdog service daemon is integrated into the SMF framework. You can change the parameters, but do not 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
```

Table 9.3, “Oracle VM VirtualBox Watchdog Configuration Parameters” also applies for Oracle Solaris. The parameter names must be changed to lowercase and a prefix of `config/` has to be added. For example: `config/user` or `config/balloon_safetymargin`. If you made any change, do not 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 the following command:

```
svcprop -p config svc:/application/virtualbox/balloonctrl:default
```

When everything is configured correctly you can start the Oracle VM VirtualBox watchdog service with the following command:

```
svcadm enable svc:/application/virtualbox/balloonctrl:default
```

For more information about SMF, please refer to the Oracle Solaris documentation.

9.20. Other Extension Packs

Another extension pack called VNC is available. This extension pack is open source and replaces the previous integration of the VNC remote access protocol. This is experimental code, and is initially available in the Oracle VM 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 with 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 Section 7.1, “Remote Display (VRDP Support)”, 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 --vrde-property 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 only as secure as using clear-text passwords.

The following command switches back to VRDP, if installed:

```
VBoxManage setproperty vrdeextpack "Oracle VM VirtualBox Extension Pack"
```

9.21. Starting Virtual Machines During System Boot

You can start VMs automatically during system boot on Linux, Oracle Solaris, and macOS platforms for all users.

9.21.1. Linux: Starting the Autostart Service With 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
}
```

Any user who wants to enable autostart for individual machines must set the path to the autostart database directory with the following command:

```
VBoxManage setproperty autostartdbpath autostart-directory
```

9.21.2. Oracle Solaris: Starting the Autostart Service With SMF

On Oracle Solaris hosts, the Oracle VM VirtualBox autostart daemon is integrated into the SMF framework. To enable it you must point the service to an existing configuration file which has the same format as on Linux, see Section 9.21.1, “Linux: Starting the Autostart Service With init”. For example:

```
# svccfg -s svc:/application/virtualbox/autostart:default setprop \
  config/config=/etc/vbox/autostart.cfg
```

When everything is configured correctly you can start the Oracle VM VirtualBox autostart service with the following command:

```
# svcadm enable svc:/application/virtualbox/autostart:default
```

For more information about SMF, see the Oracle Solaris documentation.

9.21.3. macOS: Starting the Autostart Service With launchd

On macOS, launchd is used to start the Oracle VM 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 Section 9.21.1, "Linux: Starting the Autostart Service With init".

To manually start the service use the following command:

```
# launchctl load /Library/LaunchDaemons/org.virtualbox.vboxautostart.plist
```

For additional information on how launchd services can be configured see:

http://developer.apple.com/mac/library/documentation/MacOSX/Conceptual/BPSystemStartup/BPSystemStartup.html.

9.21.4. Windows: Starting the Autostart Service

On Windows, autostart functionality consist of two components. The first component is a configuration file where the administrator can both set a delayed start for the VMs and temporarily disable autostarting for a particular user. The configuration file should be located in a folder accessible by all required users but it should have permissions allowing only reading by everyone but administrators. The configuration file contains several options. The `default_policy` controls whether the autostart service allows or denies starting of a VM for users that 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
}
```

The user name can be specified using the following forms: "user", "domain\user", ".\user" and "user@domain". An administrator must add the `VBOXAUTOSTART_CONFIG` environment variable into system variables containing the path to the configuration file described above. The environment variable tells the autostart services which configuration file is used.

The second component of autostart functionality is a Windows service. Every instance of this works on behalf of a particular user using their credentials.

To enable autostarting for a particular user, a member of the administrators group must run the following command:

```
VBoxAutostartSvc install --user=user [--password-file=password_file]
```

The password file should contain the password followed by a line break. The rest of the file is ignored. The user will be asked for a password if the password file is not specified.

To disable autostarting for particular user, a member of the administrators group must run the following command:

```
VBoxAutostartSvc delete --user=user
```

If a user has changed their password then a member of the administrators group must either reinstall the service or change the service credentials using Windows Service Manager. Due to Windows security policies, the autostart service cannot be installed for users with empty passwords.

Finally, the user should define which VMs should be started at boot. The user should run the following command for every VM they wish to start at boot:

```
VBoxManage modifyvm VM name or UUID --autostart-enabled on
```

The user can remove a particular VM from the VMs starting at boot by running the following command:

```
VBoxManage modifyvm VM name or UUID --autostart-enabled off
```

> **Note**
>
> On Windows hosts, starting VMs via the autostart service might cause some issues, as the virtual machines are starting within the same session as VBoxSVC. For more information see Section 9.35, "VBoxSVC running in Windows

Session 0".

9.22. Encryption of VMs

Oracle VM VirtualBox enables you to transparently encrypt the VM data stored in the configuration file, saved state, and EFI boot data for the guest.

Oracle VM VirtualBox uses the AES algorithm in various modes. The selected mode depends on the encrypting component of the VM. Oracle VM VirtualBox supports 128-bit or 256-bit data encryption keys (DEK). The DEK is stored encrypted in the VM configuration file and is decrypted during VM startup.

Since the DEK is stored as part of the VM configuration file, it is important that the file is kept safe. Losing the DEK means that the data stored in the VM is lost irrecoverably. Having complete and up to date backups of all data related to the VM is the responsibility of the user.

The VM, even if it is encrypted, may contain media encrypted with different passwords. To deal with this, the password for the VM has a password identifier, in the same way as passwords for media. The password ID is an arbitrary string which uniquely identifies the password in the VM and its media. You can use the same password and ID for both the VM and its media.

9.22.1. Limitations of VM Encryption

There are some limitations the user needs to be aware of when using this feature:

- Exporting appliances containing an encrypted VM is not possible, because the OVF specification does not support this. The VM is therefore decrypted during export.
- The DEK is kept in memory while the VM is running to be able to encrypt and decrypt VM data. 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 VM, the password is passed in clear text using the Oracle VM VirtualBox API. This needs to be kept in mind, especially when using third party API clients which make use of the web service where the password might be transmitted over the network. The use of HTTPS is mandatory in such a case.

9.22.2. Encrypting a VM

Encrypting a VM can be done either using VirtualBox Manager or the **VBoxManage**. To encrypt an unencrypted VM with **VBoxManage**, use:

```
VBoxManage encryptvm uuid|vmname setencryption --new-password filename|- \
--cipher cipher-ID --new-password-id ID
```

To supply the encryption password, point **VBoxManage** to the file where the password is stored or specify `-` to let **VBoxManage** prompt for the password on the command line.

The cipher parameter specifies the cipher to use for encryption and can be either `AES-128` or `AES-256`. The appropriate mode of operation, such as GCM, CTR, or XTS will be selected by the VM depending on the encrypting component. The specified password identifier can be freely chosen by the user and is used for correct identification when supplying multiple passwords for the VM.

9.22.3. Opening the Encrypted VM

When Oracle VM VirtualBox has just started up the encrypted VM cannot be opened and it stays inaccessible. Also, the encrypted VM stays inaccessible if it was just registered without a password or the password is incorrect. The user needs to provide the password using VirtualBox Manager or with the following **VBoxManage** command:

```
VBoxManage encryptvm uuid|vmname addpassword --password filename|- --password-id ID
```

To supply the encryption password point **VBoxManage** to the file where the password is stored or specify `-` to let **VBoxManage** prompt for the password on the command line.

If *ID* is the same as the password identifier supplied when encrypting the VM it updates the accessibility state.

To remove the entered password from the VM memory, use **VBoxManage** as follows:

```
VBoxManage encryptvm uuid|vmname removepassword ID
```

If *ID* is the same as the password identifier supplied when encrypting the VM it updates the accessibility state.

Note

If a machine becomes inaccessible all passwords are purged. You have to add required passwords again, using the **VBoxManage encryptvm** *vmname* **addpassword** command. See Section 9.22.3, "Opening the Encrypted VM".

9.22.4. Decrypting Encrypted VMs

In some circumstances it might be required to decrypt previously encrypted VMs. This can be done in VirtualBox Manager or using **VBoxManage** with the following command:

```
VBoxManage encryptvm uuid|vmname setencryption --old-password file|-
```

The only required parameter is the password the VM was encrypted with. The options are the same as for encrypting VMs.

9.23. Oracle VM VirtualBox Expert Storage Management

In case the snapshot model of Oracle VM 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
```

You can reconfigure storage attachments later while the VM is paused by using the **VBoxManage storageattach** command.

9.24. Handling of Host Power Management Events

Some host power management events are handled by Oracle VM 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, Oracle VM 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, Oracle VM 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, Oracle VM 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.

9.25. 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. The instruction sets are exposed to guests by default, but it is possible to disable the instructions for certain guests by using the following commands:

```
$ VBoxManage setextradata VM-name \
VBoxInternal/CPUM/IsaExts/SSE4.1 0
$ VBoxManage setextradata VM-name \
VBoxInternal/CPUM/IsaExts/SSE4.2 0
```

These are per-VM settings which are enabled by default.

9.26. Support for Keyboard Indicator Synchronization

This feature makes the host keyboard indicators (LEDs) match those of the VM's emulated keyboard when the machine window is active. It is currently implemented for macOS and Windows hosts. This feature is enabled by default on supported host OSes. You can disable this feature by running the following command:

```
$ VBoxManage setextradata VM-name GUI/HidLedsSync 0
```

This is a per-VM setting that is enabled by default.

9.27. Capturing USB Traffic for Selected Devices

You can 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. Oracle VM 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.

9.28. Configuring the Heartbeat Service

Oracle VM 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 Oracle VM VirtualBox Guest Additions will send a heartbeat ping every two seconds. The value in nanoseconds can be configured like this:

```
VBoxManage setextradata 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 setextradata 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.*

9.29. Encryption of Disk Images

Oracle VM VirtualBox enables you to transparently encrypt the data stored in hard disk images 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 Oracle VM VirtualBox and other virtualization software.

Oracle VM VirtualBox uses the AES algorithm in XTS mode and supports 128-bit 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.

9.29.1. Limitations of Disk Encryption

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 does not 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 using the Oracle VM 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 Oracle VM VirtualBox version.
- The disk encryption feature can protect the content of the disks configured for a VM only. It does not cover any other data related to a VM, including saved state or the configuration file itself.

9.29.2. Encrypting Disk Images

Encrypting disk images can be done either using VirtualBox Manager or the **VBoxManage**. While VirtualBox Manager 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 filename|- --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.

9.29.3. Starting a VM with Encrypted Images

When a VM is started using VirtualBox Manager, 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]
```

ID must be the same as the password identifier supplied when encrypting the images. *password* is the password used when encrypting the images. Optionally, you can 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 does not want the password to remain in memory.

9.29.4. Decrypting Encrypted Images

In some circumstances it might be required to decrypt previously encrypted images. This can be done in VirtualBox Manager 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.

9.30. Paravirtualized Debugging

This section covers 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.

9.30.1. 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 by 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 by 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.

9.30.1.1. 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. USB and IEEE 1394 are not applicable for virtual machines, and Ethernet requires Windows 8 or later. While a 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 can check the effective paravirtualization provider for your VM with 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. 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.

  > **Note**
  >
  > 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 with the following **VBoxManage** command:

   ```
   $ VBoxManage modifyvm VM-name --paravirt-debug "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. For example:

   ```
   $ VBoxManage modifyvm VM-name \
   --paravirt-debug "enabled=1,address=192.168.32.1,port=55000"
   ```

   See Section 9.30.1, “Hyper-V Debug Options” 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.

   - Run 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**. On the **NET** tab, specify the UDP port number you used in the `paravirtdebug` options. If you did not 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.

   Click **OK** to start listening for connections. Microsoft WinDbg typically shows a Waiting to Reconnect message during this phase.

   Alternatively, to directly start a debug session, run WinDbg from the command line as follows :

   ```
   windbg.exe -k net:port=50000,key=1.2.3.4
   ```

   See the WinDbg documentation for the complete command line syntax.

6. Reboot your Windows guest and it should then connect as a debuggee with Microsoft WinDbg.

9.31. PC Speaker Passthrough

As an experimental feature, primarily due to being limited to Linux host only and unknown Linux distribution coverage, Oracle VM VirtualBox supports passing through the PC speaker to the host. The PC speaker, sometimes called the system speaker, is a way to produce audible feedback such as beeps without the need for regular audio and sound card support.

The PC speaker passthrough feature in Oracle VM 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 a very complex topic. Oracle VM 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. These are summarized in the following table.

Table 9.4. PC Speaker Configuration Options

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 `cap_sys_tty_config` capability.
9	A user-specified console or evdev device path.	As for codes 1 to 3, but with a custom device path.
70	`/dev/tty`	Standard beep only. Loses frequency and length. See code 2.
79	A user-specified terminal device path.	As for code 70, but with a custom device path.
100	All of the above.	Tries all the available 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 takes effect when you next start the VM. 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`, contains 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 Oracle VM 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, such as 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. For example, 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 Oracle VM VirtualBox. The table above contains some hints, but in general 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", which could be hidden in the mixer settings, and that it is not muted.

9.32. Accessing USB devices Exposed Over the Network with USB/IP

Oracle VM 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 Oracle VM 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, use the following command to add the device server that exports the devices:

```
VBoxManage usbdevsource add unique-name --backend USBIP --address device-server[:port]
```

USB devices exported on the device server are then accessible through VirtualBox Manager 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
```

9.32.1. 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, use the following command 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
```

This 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
```

See 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. This must be done manually for each device. To export a device use the following command:

```
# usbip bind -b "bus identifier"
```

To export the multicard reader in the previous example:

```
# usbip bind -b 4-2
```

9.32.2. 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.

9.33. Using Hyper-V with Oracle VM VirtualBox

Oracle VM VirtualBox can be used on a Windows host where Hyper-V is running. This is an experimental feature.

No configuration is required. Oracle VM VirtualBox detects Hyper-V automatically and uses Hyper-V as the virtualization engine for the host system. The CPU icon in the VM window status bar indicates that Hyper-V is being used.

> **Note**
>
> When using this feature, some host systems might experience significant Oracle VM VirtualBox performance degradation.

9.34. Nested Virtualization

Oracle VM VirtualBox supports *nested virtualization*. This feature enables the passthrough of hardware virtualization functions to the guest VM. That means that you can install a hypervisor, such as Oracle VM VirtualBox, Oracle VM Server or KVM, on an Oracle VM VirtualBox guest. You can then create and run VMs within the guest VM.

Hardware virtualization features not present on the host CPU will not be exposed to the guest. In addition, some features such as nested paging are not yet supported for passthrough to the guest.

You can enable the nested virtualization feature in one of the following ways:

- From VirtualBox Manager, select the **Enable Nested VT-x/AMD-V** check box on the **Processor** tab. To disable the feature, deselect the check box.
- Use the `--nested-hw-virt` option of the **VBoxManage modifyvm** command to enable or disable nested virtualization. See Section 8.10, “VBoxManage modifyvm”.

9.35. VBoxSVC running in Windows Session 0

Oracle VM VirtualBox supports executing the VBoxSVC in Windows session 0. This allows VBoxSVC to run like a regular Windows service, which in turn enables headless VMs to continue running even if the user logs out.

> **Note**
>
> This is currently an experimental feature.

The feature is disabled by default and can be enabled by creating a REG_DWORD value `ServerSession0` in the key `HKEY_LOCAL_MACHINE\Software\Oracle\VirtualBox\VBoxSDS` of the Windows registry. Specify `1` as the value's data to enable the feature, or `0` to disable the feature. A host reboot is needed in order to make the change effective.

9.35.1. Known Issues

- Due to different Windows sessions having their own set of resources, there might be some issues with accessing network shares created in the interactive user session when at least one of the Oracle VM VirtualBox processes are running in session 0.

  For accessing network shares within session 0, a possible workaround is to establish permanent access to the share and then restart the host.

9.36. VISO file format / RTIsoMaker

ISO image maker.

Synopsis

```
RTIsoMaker [options] [@commands.rsp] <filespec...>
```

Description

Construct a virtual ISO 9660 / Joliet / UDF / HFS hybrid image and either write it to a file (RTIsoMaker) or serve it as a virtual image (VISO).

VISO file format

A VISO file is a virtual ISO image, i.e. constructed in memory from a bunch of files on the host. A VISO is just the recipe describing how to go about this using a syntax vaguely similar to mkisofs and genisoimage.

One requirement is that the VISO file must start with one of the `--iprt-iso-maker-file-marker` options. Which of the options you use will dictate the quoting and escaping rules used when reading the file. The option takes the image UUID as an argument.

The VISO files are treated as UTF-8 and must not contain any byte order marker (BOM). There is currently no way to comment out lines in a VISO file.

File specifications and --name-setup

All non-options that does not start with '@' are taken to indicate a file, directory, or similar that is should be added to the ISO image. Directories are added recursively and content is subject to filtering options.

Since there can be up to six different namespaces on an ISO, it is handy to be able to control the names used in each and be able to exclude an object from one or more namespaces. The `--name-setup` option specifies the file specification format to use forthwith.

The default setup is:

```
--name-setup iso+joliet+udf+hfs
```

Which means you specify one on-ISO name for all namespaces followed by '=' and the source file system name. Only specifying the source file system will add the file/dir/whatever to the root of the ISO image.

Lets look at the following two examples:

```
/docs/readme.txt=/home/user/Documents/product-x-readme.txt
```

```
/home/user/Documents/product-x-readme.txt
```

In the first case the file `'/home/user/Documents/product-x-readme.txt'` is added to the ISO image as `'/docs/readme.txt'` in all enabled namespaces. In the primary ISO 9660 namespace, the filename will by default be converted to upper case because it's required by the spec.

In the second case the file is added to the root under the name `'product-x-readme.txt'` in all namespaces. Though, in the primary ISO 9660 namespace the name will be transformed to apply with the current ISO level, probably uppercased, possibly truncated too.

Given `--name-setup iso,joliet,udf` you can specify the name individually for each of the three namespace, if you like. If you omit any, they will use last name given. Any names left blank (`==`) will be considered omitted.

A different name in each namespace:

```
/ISO.TXT=/Joliet.TxT=/UDF.txt=/tmp/iso/real.txt
```

Specific name in the ISO 9660 namespace, same in the rest:

```
/ISO.TXT=/OtherNamespaces.TxT=/tmp/iso/real.txt
```

Omit the file from the ISO 9660 namespace:

```
=/OtherNamespaces.TxT=/tmp/iso/real.txt
```

Omit the file from the joliet namespace:

```
/ISO.TXT==/UDF.TxT=/tmp/iso/real.txt
```

Use the same filename as the source everywhere:

```
/tmp/iso/real.txt
```

Using for instance `--name-setup udf` you can add a files/dirs/whatever to select namespace(s) without the more complicated empty name syntax above.

When adding directories, you can only control the naming and omitting of the directory itself, not any recursively added files and directories below it.

Options

General

`-o output-file`, `--output=output-file`

The output filename. This option is not supported in VISO mode.

`--name-setup=spec`

Configures active namespaces and how file specifications are to be interpreted. The specification is a comma separated list. Each element in the list is a sub-list separated by space, `'+'` or `'|'` giving the namespaces that elements controls. Namespaces are divied into two major and minor ones, you cannot specifying a minor before the major it belongs to.

Major namespaces and aliases in parentheses:

- iso (primary, iso9660, iso-9660, primary-iso, iso-primary)
- joliet
- udf
- hfs (hfs-plus)

Minor namespaces:

- rock: rock ridge on previous major namespace (iso / joliet)
- iso-rock: rock ridge extensions on primary ISO 9660 namespace
- joliet-rock: rock ridge on joliet namespace (just for fun)
- trans-tbl: translation table file on previous major namespace
- iso-trans-tbl
- joliet-trans-tbl
- udf-trans-tbl
- hfs-trans-tbl

`--name-setup-from-import`

This is for use following one or more `--import-iso` operations and will pick a configuration matching the imported content as best we can. If the imported ISOs only had a iso9660 namespace, the joliet, udf and hfs namespaces will be removed. This is useful when adding additional files to the ISO and will prevent guest from picking a namespace without the imported ISO content when mounting it.

`--push-iso=iso-file`, `--push-iso-no-joliet=iso-file`, `--push-iso-no-rock-iso-file`, `--push-iso-no-rock-no-joliet=iso-file`

Open the specified ISO file and use it as source file system until the corresponding `--pop` options is encountered. The variations are for selecting which namespace on the ISO to (not) access. These options are handy for copying files/directories/stuff from an ISO without having to extract them first or using the `:iprtvfs:` syntax.

`--pop`

Pops a `--push-iso` of the source file system stack.

`--import-iso=iso-file`

Imports everything on the given ISO file, including boot configuration and system area (first 16 sectors) content. You can use `--name-setup` to omit namespaces.

Namespaces

`--iso-level=0|1|2|3`

Sets the ISO level:

- 0: Disable primary ISO namespace.

- 1: ISO level 1: Filenames 8.3 format and limited to 4GB - 1.
- 2: ISO level 2: 31 char long names and limited to 4GB - 1.
- 3: ISO level 3: 31 char long names and support for >=4GB files. (default)
- 4: Fictive level used by other tools. Not yet implemented.

`--rock-ridge`, `--limited-rock-ridge`, `--no-rock-ridge`

Enables or disables rock ridge support for the primary ISO 9660 namespace. The `--limited-rock-ridge` option omits a couple of bits in the root directory that would make Linux pick rock ridge over joliet.

Default: `--limited-rock-ridge`

`-J`, `--joliet`, `--no-joliet`

Enables or disable the joliet namespace. This option must precede any file specifications.

Default: `--joliet`

`--joliet-ucs-level=`*`1`*|*`2`*|*`3`*, `--ucs-level=`*`1`*|*`2`*|*`3`*

Set the Joliet UCS support level. This is currently only flagged in the image but not enforced on the actual path names.

Default level: 3

File Attributes

`--rational-attribs`

Enables rational file attribute handling (default):

- Owner ID is set to zero
- Group ID is set to zero
- Mode is set to 0444 for non-executable files.
- Mode is set to 0555 for executable files.
- Mode is set to 0555 for directories, preserving stick bits.

`--strict-attribs`

Counters `--rational-attribs` and causes attributes to be recorded exactly as they appear in the source.

`--file-mode=`*`mode`*, `--no-file-mode`

Controls the forced file mode mask for rock ridge, UDF and HFS.

`--dir-mode=`*`mode`*, `--no-dir-mode`

Controls the forced directory mode mask for rock ridge, UDF and HFS.

`--new-dir-mode=`*`mode`*

Controls the default mode mask (rock ridge, UDF, HFS) for directories that are created implicitly. The `--dir-mode` option overrides this.

`--chmod=`*`mode`*`:`*`on-iso-file`*

Explictily sets the rock ridge, UDF and HFS file mode for a file/dir/whatever that has already been added to the ISO. The mode can be octal, `ra+x`, `a+r`, or `a+rx`. (Support for more complicated mode specifications may be implemented at a later point.)

Note that only namespaces in the current --name-setup are affected.

`--chown=`*`owner-id`*`:`*`on-iso-file`*

Explictily sets the rock ridge, UDF and HFS file owner ID (numeric) for a file/dir/whatever that has already been added to the ISO.

Note that only namespaces in the current --name-setup are affected.

--chgrp=*group-id*:*on-iso-file*

Explictily sets the rock ridge, UDF and HFS file group ID (numeric) for a file/dir/whatever that has already been added to the ISO.

Note that only namespaces in the current --name-setup are affected.

Booting

`--eltorito-new-entry`, `--eltorito-alt-boot`

Starts a new El Torito boot entry.

`--eltorito-add-image=`*filespec*

File specification of a file that should be added to the image and used as the El Torito boot image of the current boot entry.

`-b` *on-iso-file*, `--eltorito-boot=`*on-iso-file*

Specifies a file on the ISO as the El Torito boot image for the current boot entry.

`--eltorito-floppy-12`, `--eltorito-floppy-144`, `--eltorito-floppy-288`, `--no-emulation-boot`, `--hard-disk-boot`

Sets the boot image emulation type of the current El Torito boot entry.

`--boot-load-seg=`*seg*

Specify the image load segment for the current El Torito boot entry.

Default: 0x7c0

`--boot-load-size=`*sectors*

Specify the image load size in emulated sectors for the current El Torito boot entry.

Default: 4 (sectors of 512 bytes)

`--no-boot`

Indicates that the current El Torito boot entry isn't bootable. (The BIOS will allegedly configure the emulation, but not attempt booting.)

`--boot-info-table`

Write a isolinux/syslinux boot info table into the boot image for the current El Torito boot entry.

`--eltorito-platform-id=`*id*

Set the El Torito platform ID of the current entry, a new entry of the verification entry depending on when it's used. The ID must be one of: `x86`, `PPC`, `Mac`, `efi`

`-c` *namespec*, `--boot-catalog=`*namespec*

Enters the El Torito boot catalog into the namespaces as a file. The *namespec* uses the same format as a 'filespec', but omits the final source file system name component.

`-G` *file*, `--generic-boot=`*file*

Specifies a file that should be loaded at offset 0 in the ISO image. The file must not be larger than 32KB. When creating a hybrid image, parts of this may be regenerated by partition tables and such.

String properties (applied to active namespaces only)

`--abstract=`*file-id*

The name of the abstract file in the root dir.

`-A` *text*|*_file-id*, `--application-id=`*text*|*_file-id*

Application ID string or root file name. The latter must be prefixed with an underscore.

`--biblio=`*file-id*

The name of the bibliographic file in the root dir.

`--copyright=`*file-id*

The name of the copyright file in the root dir.

`-P` *text*|*_file-id*`, --publisher=`*text*|*_file-id*

Publisher ID string or root file name. The latter must be prefixed with an underscore.

`-p` *text*|*_file-id*`, --preparer=`*text*|*_file-id*

Data preparer ID string or root file name. The latter must be prefixed with an underscore.

`--sysid=`*text*

System ID string.

`--volid=`*text*`, --volume-id=`*text*

Volume ID string (label). (It is possible to set different labels for primary ISO 9660, joliet, UDF and HFS by changing the active namespaces using the `--name-setup` option between `--volume-id` occurences.)

`--volset=`*text*

Volume set ID string.

Compatibility:

`--graft-points`

Alias for --name-setup iso+joliet+udf+hfs.

`-l`, `--long-names`

Allow 31 charater filenames. Just ensure ISO level >= 2 here.

`-R`, `--rock`

Same as `--rock-ridge` and `--strict-attribs`.

`-r`, `--rational-rock`

Same as `--rock-ridge` and `--rational-attribs`.

VISO Specific:

`--iprt-iso-maker-file-marker=`*UUID*`, --iprt-iso-maker-file-marker-bourne=`*UUID*`, --iprt-iso-maker-file-marker-bourne-sh=`*UUID*

Used as first option in a VISO file to specify the file UUID and that it is formatted using bourne-shell argument quoting & escaping style.

`--iprt-iso-maker-file-marker-ms=`*UUID*`, --iprt-iso-maker-file-marker-ms-sh=`*UUID*

Used as first option in a VISO file to specify the file UUID and that it is formatted using microsoft CRT argument quoting & escaping style.

Testing (not applicable to VISO):

`--output-buffer-size=`*bytes*

Selects a specific output buffer size for testing virtual image reads.

`--random-output-buffer-size`

Enables randomized buffer size for each virtual image read, using the current output buffer size (`--output-buffer-size`) as maximum.

`--random-order-verification=`*size*

Enables verification pass of the image that compares blocks of the given size in random order from the virtual and output images.