acontis technologies GmbH

SOFTWARE

Hypervisor

User's Manual

Version 9.x

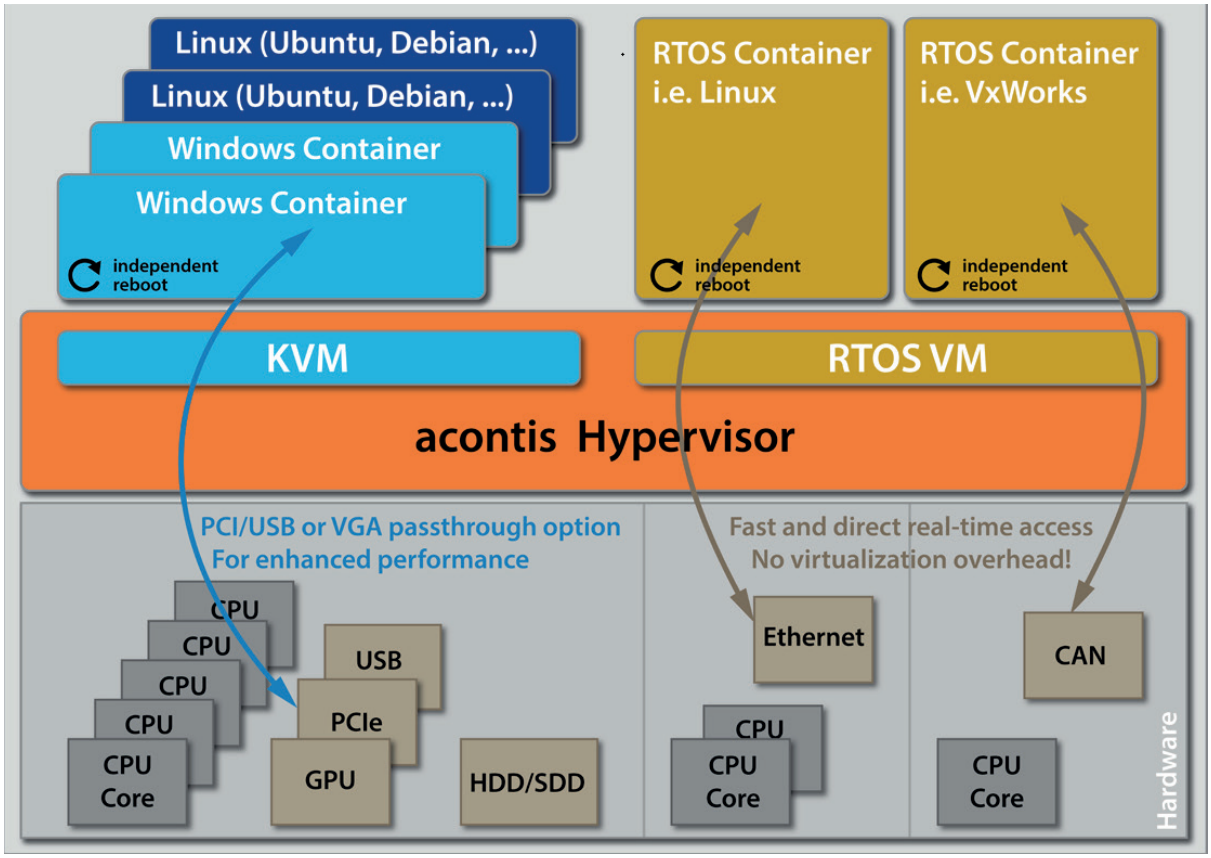
Edition: July 29, 2025

Contents

1 Introduction

1.1 Overview

Using the existing, industry proven acontis real-time RTOS-VM virtualization technology, multiple hard real-time operating systems (Real-time Linux, VxWorks, etc.) can run in native speed. In addition, based on the well-known and proven KVM virtualization solution, multiple standard operating systems like Windows and Linux (Ubuntu, Debian, Fedora, etc.) operate in parallel. Besides virtual hardware, KVM provides para-virtualization, PCI/USB and VGA pass-through for highest possible performance. Each guest OS is fully independent and separated and can be rebooted or shutdown while the other guests continue without being affected. The Hypervisor Host (or service OS) is based on a stripped down Debian Linux operating system.



RTOS Virtual Machine Hypervisor

The RTOS Virtual Machine hypervisor technology provides an independent layer to run any RTOS in native speed. No virtualization overhead is introduced and all RTOS drivers as well the operating systems and applications have direct and fast hardware access. The same technology is successfully used by customers all over the world since more than 10 years in the existing acontis real-time solution. The new acontis Hypervisor can utilize this technology without modification so existing customer applications can be re-used without modification.

KVM Hypervisor

KVM is one of the most popular Type 1 hypervisors used in many high-availability and security critical

environments like the cloud solutions from Google, Amazon or Oracle.

To increase performance, for example for fast USB, Ethernet or Graphics operation, the respective devices can be completely passed through to Windows or Linux guests. Alternatively, para-virtualized devices for the hard disk, the Ethernet or graphics controller reduce the amount of necessary hardware without compromising throughput.

Key technical features

The acontis Hypervisor is a perfect symbiosis between the wide-spread KVM virtualization solution and the industry proven RTOS Virtual Machine hypervisor for real-time operating systems.

General

- Supports Multiple OSes: real-time Linux, On Time RTOS-32, VxWorks® RTOS, Standard Linux, Microsoft® Windows®, proprietary Roll-your-own, Bare metal, any unmodified OS
- RTOS containers including applications run on bare metal core with no virtualization overhead and direct hardware access
- Fully separated and independent guest operation
- User defined guest startup sequence
- Utilize any number of CPU cores per single guest
- Independent reboot of all guests while others continue operation
- Virtual Network between all guests
- Inter-OS Communication: Shared Memory, Events, Interlocked data access, Pipes, Message Queues and Real-time sockets for high speed application level communication
- Hypervisor provided fileserver for all guests

KVM

Windows and Standard Linux operating systems like Ubuntu or Debian run under control of the KVM hypervisor. This hypervisor provides plenty of sophisticated features:

- Multiple Windows and/or standard Linux instances
- Windows/Linux containers with snapshot support to easily switch between different application situations without the need to install multiple OS instances. Snapshots create a view of a running virtual machine at a given point in time. Multiple snapshots can be saved and used to return to a previous state, in the event of a problem.
- Pass-through support: To increase performance, for example for fast USB, Ethernet or Graphics operation, the respective devices can be completely passed through to Windows or Linux guests.
- Paravirtualization: Para-virtualized devices for the hard disk, the ethernet or graphics controller reduce the amount of necessary hardware without compromising throughput.
- Graphics virtualization to provide 3D accelerated graphics to multiple guests.

RTOS-VM

- Multiple real-time Operating Systems (Linux, VxWorks, On Time RTOS-32, etc.)
- Fast real-time interrupt handling and short thread latencies
- Direct hardware access with no virtualization overhead

- Compatible with the existing acontis Windows real-time extension (applications can be shared or cross-migrated between both solutions)

1.2 Linux tips for Windows users

The following points may help a Windows user with the first Linux steps.

- Windows Command Prompt is called `Terminal` in Linux. The shortkey to open one is `Ctrl` + `Alt` + `T`.
- Instead of `notepad.exe` you can use `mousepad`. If you have no graphical desktop available `nano` will work.
- `cd \MyDir\MySubDir` is `cd /MyDir/MySubDir` and `cd..` must be `cd ...`
- `dir` is `ls` and `ls -l` will show additional information.
- `run as administrator` has its equivalent in `sudo`, which has to be put before a command. For example `sudo mousepad` starts the editor as administrator (called `root` in Linux). You can use `sudo -s` to switch console to root user and `exit` to return.
- Most programs give helpful information when called with parameter `--help` or by calling the manual `man`. In case of `mousepad` this would be `mousepad --help` and `man mousepad`.
- `chmod` can change the rights (*read/write/executable*) of a file. In Windows you just create a file `test.bat` and you can already execute it. In Linux you name it `test.sh`, but it won't run without being made an `executable` (i.e. `chmod +x test.sh`).
- `chown` can change the owner of a file.

2 System Setup

2.1 Hardware Requirements

Ressource	Recommended
CPU	Minimum: • `1` Core for the Hypervisor Host, which can be shared with non-real-time OS • `+1` Core for *each* real-time OS. • `+1` Core for *each* non-real-time OS if **not** shared with Hypervisor Host. **Important:** Do **not** enable Hyperthreading. **Important:** CPU **must** support `VT` technology!
Memory	`4GB` for Hypervisor Host and RTOS + minimum `4GB` for Windows (as a guest).
GPU	
2nd GPU	
Disk Space	`50GB` for Hypervisor Host + minimum `50GB` for Windows (as a guest).

Hint: For installation purposes it is recommended to use a USB stick with at least `4GB`. Further information can be found at the Hypervisor Quickstart Guide!

2.2 Installation and Configuration

Hint: The hypervisor could be installed either exclusively on a PC or side by side with an existing OS. The **default** case is exclusively. The side by side installation is described in the next chapter *Side By Side Installation*!

2.2.1 Side By Side Installation

If the hypervisor has been installed in parallel to Windows or Linux, the file `setrootuuid.sh` has to be adjusted. The original `defaultBoot` values need to be incremented by 1, e.g. set to `3` and `3`, respectively.

Enter `mousepad setrootuuid.sh` to adjust these values, they are near the end of the file:

```
# update default boot configuration
###################################
defaultbootRE='\(GRUB_DEFAULT=\)[0-9]*'
[ -d /sys/firmware/efi ] && defaultBoot=3 \|\| defaultBoot=3 # 2 for UEFI,␣
↪2 for Legacy BIOS
```

Caution: The above numbers (2, 3) may change depending on the hypervisor version!

Restart the system by entering:

```
$ sudo reboot
```

In the next step you need to initialize the Hypervisor. See *Hypervisor Initialization* for details.

2.3 Hypervisor Files and Directories

2.3.1 Directories

Directory	SubDir	Content
/hv	.	Hypervisor root directory
	bin/.	Binaries and scripts
	config/.	General config files and configuration scripts
	doc/.	Manuals and guides
	services/.	Services (automatically started when booting)
	templates/.	Template files (for guests, configuration, …)
	sdk/.	Software Development Kit
	sysmgr/.	System Manager (graphical configuration and diagnosis tool)
/hv/guests	.	Guests root folder
	files/.	Filesystem folder to be shared between guests
	etc/.	Additional guest specific examples, tools etc.

2.3.2 Configuration Files

The hypervisor configuration files are located in `/hv/config` and in the guest folders. The initial configuration files are located in `/hv/templates/config`. They are copied to `/hv/config` when installing the Hypervisor (via the `/hv/bin/inithv.sh` initialization script).

The root configuration file `hv.config` is used by the hypervisor for the overall guest configuration. The following table shows the locations of the root configuration file `hv.config`

Directory	SubDir	Description
/hv	config/	Used by the System Manager generated guests

2.3.3 Brand Labeling

In case if you need to do brand labeling the shipped product, here is the list of image files for wallpapers, boot logos etc

Directory	SubDir	graphic file	Description
/usr/share	plymouth/themes/rtosvisor	*.png	OS Boot logo
	images/desktop-base	*.png	Logos
	images/desktop-base	*.svg	Login picture

3 Hypervisor Initialization

3.1 First Time Initialization (V8.x only)

After the installation of RTOSVisor, the boot configuration `Configure RTOSVisor` is automatically executed during the first startup and RTOSVisor is initialized. Afterwards, RTOSVisor automatically restarts.

3.2 Reset Hypervisor Initialization

At any time later, you can reset the hypervisor initialization using the `hv_resethv` command. The `hv_resethv` can only by called after booting the Hypervisor with selecting the `Configure RTOSVisor` boot menu entry.

Running this command will also reset the configuration (located in `/hv/config`).

You should preserve the previous configuration and guest settings in case you have changed them.

The `hv_resethv` command will automatically try to preserve these folders (e.g. into `/hv/config.bak`). If these preserved folder already exist, you may run `hv_resethv -force` to automatically remove the current settings without preserving them.

```
$ hv_resethv [-force]
```

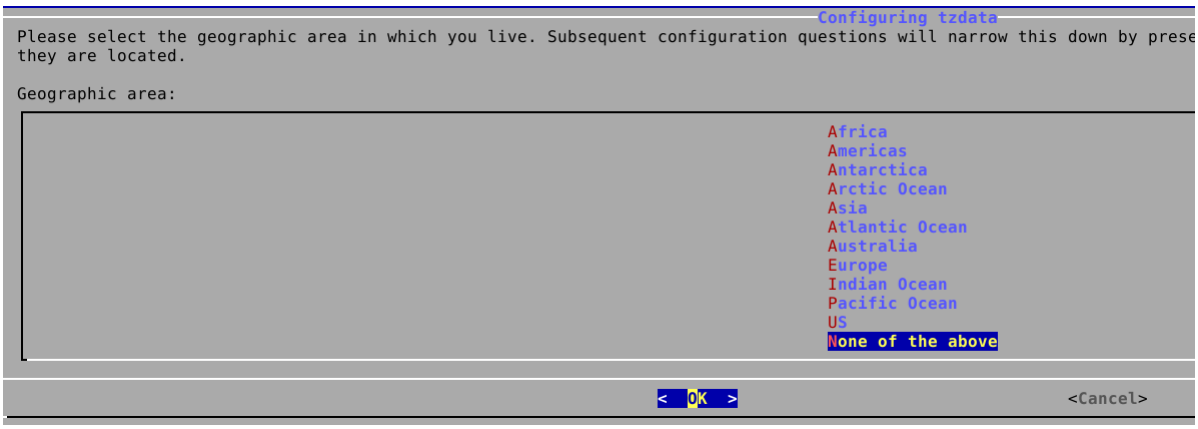
3.3 Timezone, date and time

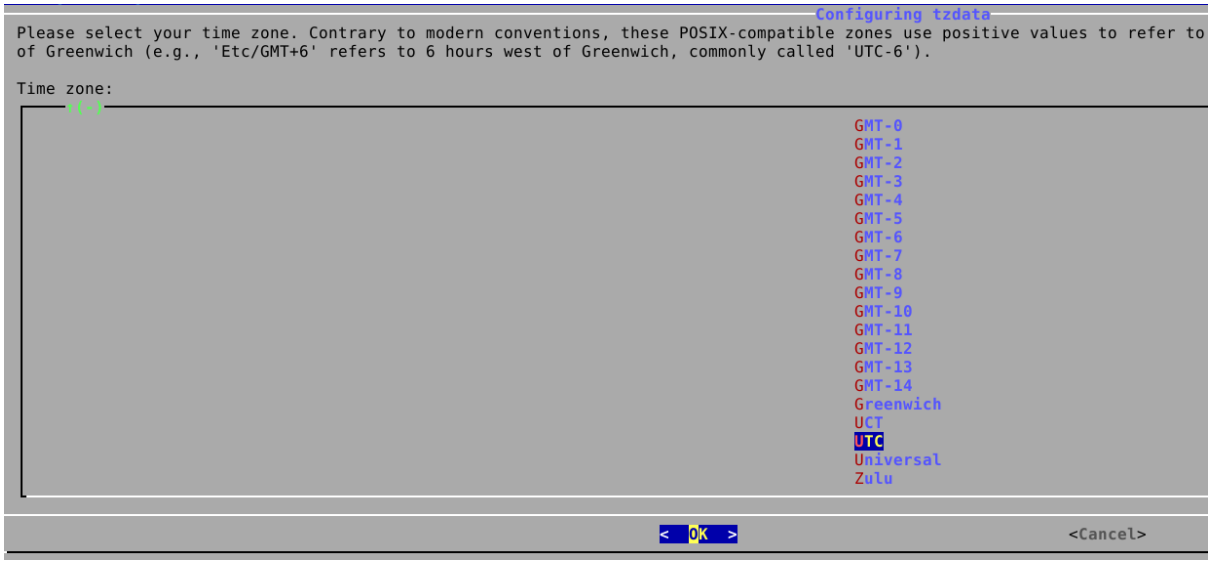
If you want to change the timezone, run the following command in a terminal window:

```
$ sudo dpkg-reconfigure tzdata
```

Then you will be prompted to select the appropriate timezone.

If you want to use UTC, follow these steps:



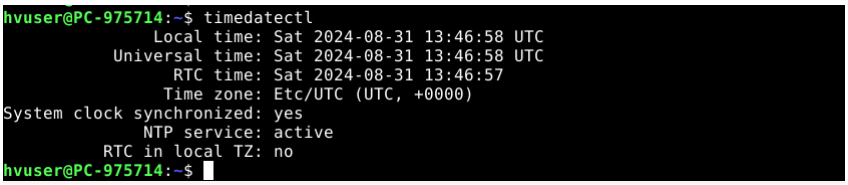


To set the date and time, use the following command:

```
$ sudo timedatectl set-time 'YYYY-MM-DD HH:MM:SS'
```

To verify the current time and date settings, run the following command:

```
$ timedatectl
```



Date and time can be synchronized using NTP (Network Time Protocol). You can turn off NTP as follows:

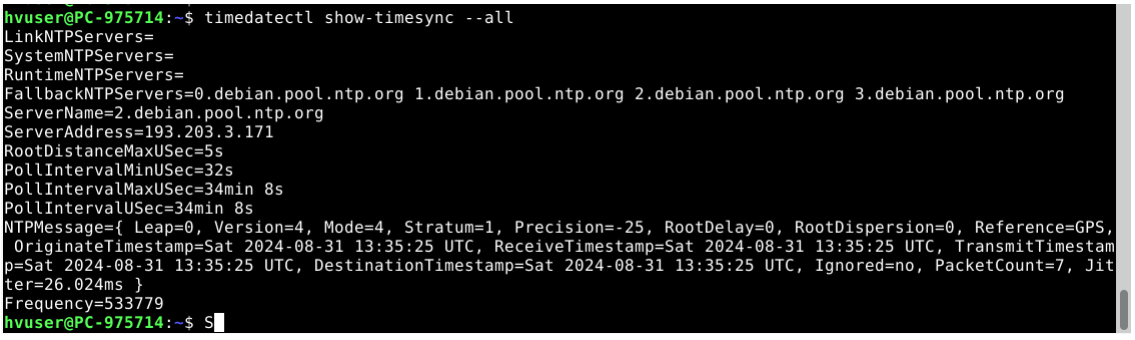
```
$ sudo timedatectl set-ntp false
```

You can turn on NTP as follows:

```
$ sudo timedatectl set-ntp true
```

You can check the NTP settings as follows:

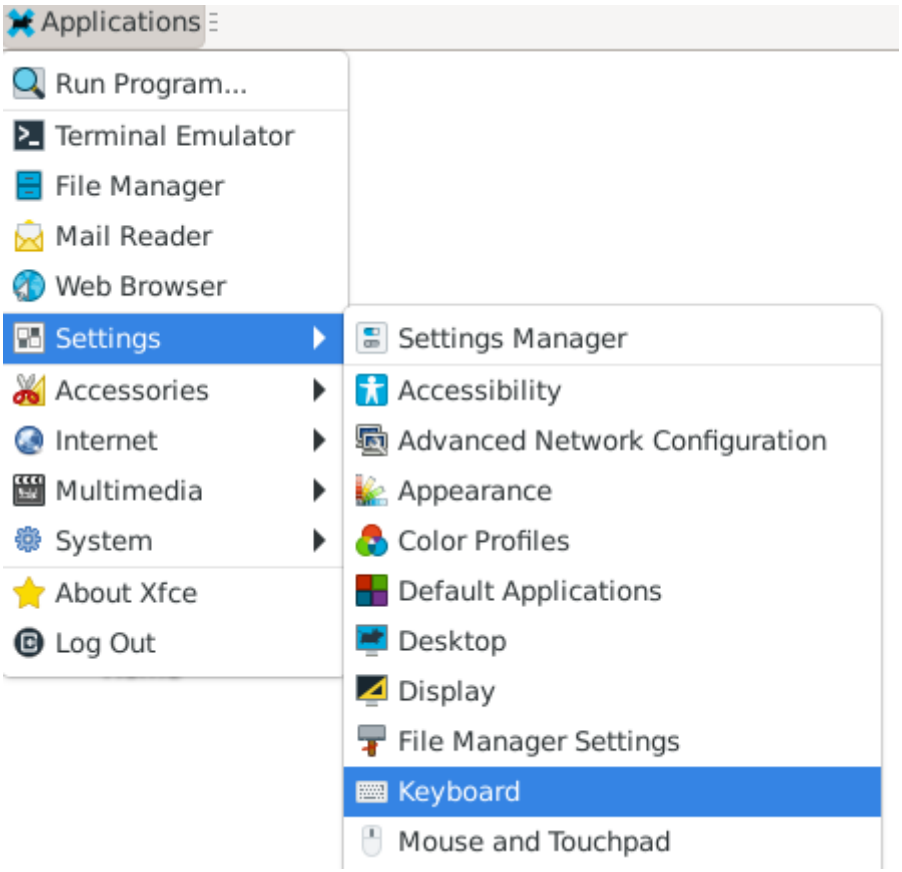
```
$ sudo timedatectl show-timesync --all
```



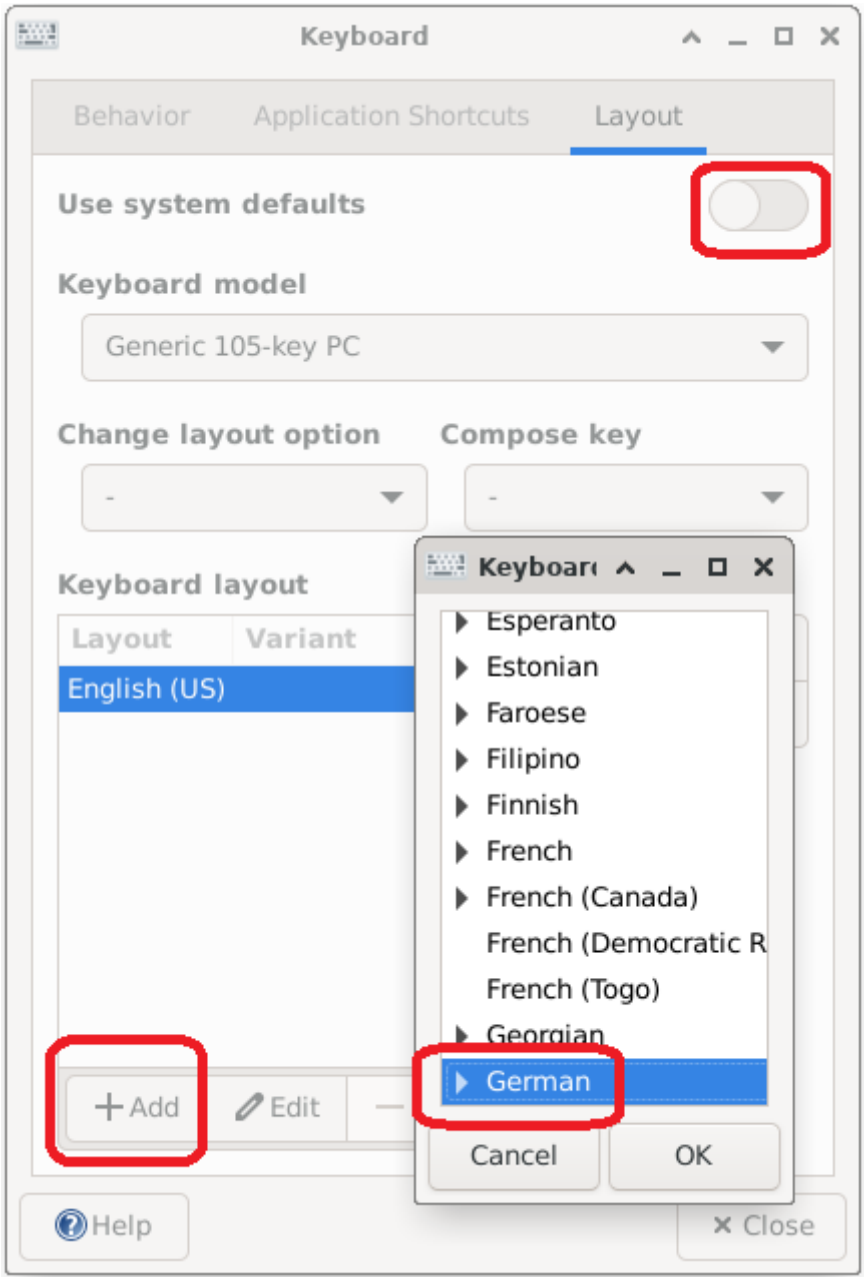
3.4 Keyboard layout (language)

The keyboard layout may be changed depending on the type of keyboard that is connected to the PC.

Use the menu to change the settings:

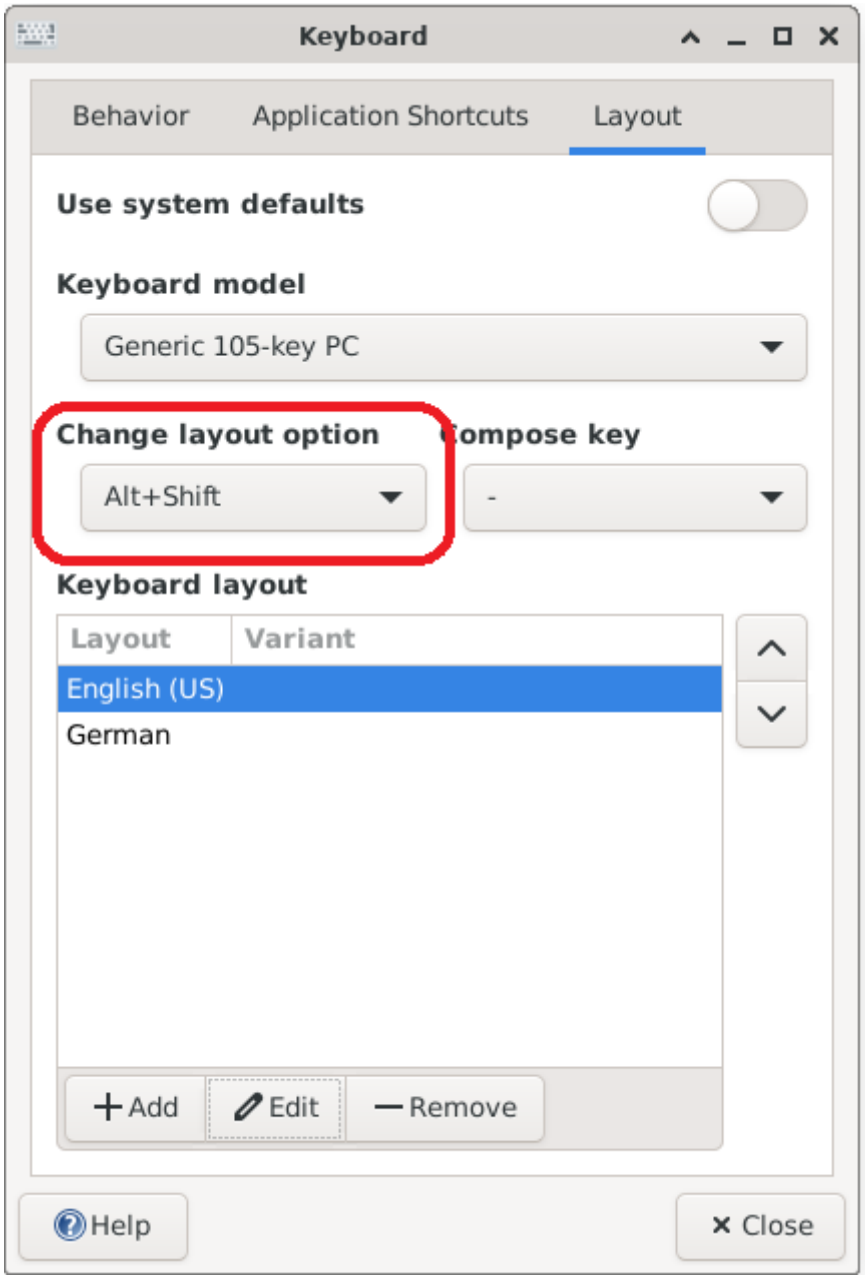


Then you need to switch to the *Layout* tab and unselect *Use system defaults*. Next, please add the new layout which fits your requirements.



You may remove layouts that you don't need or, alternatively, edit the existing layout instead of adding a new one.

Optionally, you may switch between multiple keyboard layouts using a shortcut.


Keyboard
Behavior
Application Shortcuts
Layout
Use system defaults
Keyboard model
Generic 105-key PC
Change layout option
Alt+Shift
Compose key
-
Keyboard layout
Layout
Variant
English (US)
German
Add
Edit
Remove
Help
Close

4 Hypervisor Guests - General

The RTOSVisor supports two different kinds of guest operating systems.

- Unmodified, *non* Real-time operating systems, for example Windows 10 or Ubuntu 22.04. These guests run in a virtual machine under control of the KVM hypervisor technology.
- Para-virtualized Real-time operating systems (RTOS), for example Real-time Linux. These guests use the Virtual Machine Framework (VMF) paravirtualization and run bare metal on the physical hardware.

A single guest is located in its respective guest folder.

4.1 Example guests

Some pre-configured guests are provided in RTOSVisor for test (see *Example guest folders*). It is not recommended to create new guest folders manually, for this purpose the graphical System Manager tool shall be used.

4.2 Guest filesystem access

The Hypervisor Host provides access to its filesystem (located on a physical hard disk or SSD) to guests. By default the `/hv/guests` folder is exposed to all guests.

4.3 General guest folder content

All guest folders contain the following files.

- `guest_config.sh`: the main guest configuration script, you must **not** change this file
- `usr_guest_config.sh`: user specific guest configuration script
- `guest.config`: guest specific configuration, you must **not** change this file
- `usr.config`: user specific guest configuration
- `vm_shutdown_hook.sh`: guest shutdown hook

Additional configuration files may exist depending on the guest type and operation mode.

4.4 Example guest folders

Initially there are no example guest folders on the Hypervisor Host. If you would like to switch to an example guest you have to call the depending initialization. There are several different example guest configurations you can choose:

- rt-linux : RT-Linux guest in `/hv/guests/guestrtlinux`
- rtos-32 : On Time RTOS-32 in `/hv/guests/guestrtos32`
- vxworks : VxWorks in `/hv/guests/guestvxworks`
- windows_rtlinux : windows in `/hv/guests/guestwindows` and RT-Linux guest in `/hv/guests/guestrtlinux`
- ubunutu_rtlinux : Ubuntu in `/hv/guests/guestubuntu` and RT-Linux guest in `/hv/guests/guestrtlinux`

Initialize the desired example with the commands hv_open_example, hv_sync_example and the name of the configuration. To load RT-Linux you can call

```
$ hv_open_example rt-linux
$ hv_sync_example rt-linux
$ cd /hv/guests/guestrtlinux
```

Caution: When the example guests are initialized, the corresponding pre-configured System Manager projects will be loaded. This process will delete any existing guests, subject to your confirmation. You can preserve your current configuration by using the System Manager to save it before initializing an example. Ensure you also save any manually added content in the `/hv/guests/guestxxxx` folders, as these will be deleted as well.

Each configuration is designed to operate a single RTOS. Both the Windows and Ubuntu configurations are capable of running RT-Linux simultaneously.

The Windows and Ubuntu guests are initialized without an installed operating system. For installation, the guest expects a suitable installation file in `.iso` format with the following name upon startup:

- Windows guest - `/hv/iso/windows.iso`.
- Ubuntu guest - `/hv/iso/ubuntu.iso`.

A detailed description how to transfer the `.iso` file can be found in the according Guest Guides.

During the installation of Windows or Ubuntu example guests, disk image files are generated in:

- Windows guest - `/hv/VMs/example_win/windows.qcow2`
- Ubuntu guest - `/hv/VMs/example_ubuntu/ubuntu.qcow2`

Caution: If in your own projects no `Disk Image File` is specified, then the image will be created in the Guest-Folder, and if the guest is deleted, the image and thus the entire installation and all settings in the Windows or Ubuntu guest will also be deleted.

Hint: The hypervisor uses the configuration stored in `/hv/config/hv.config`. This file is read and written by the System Manager configuration tool. Editing this file manually is not recommended.

4.5 Guest operation

To operate with a specific guest, you need to initialize the desired example configuration as described in *Example guest folders*. Then switch into the guest folder first. The following shows how to select the shipped Real-time Linux example guest.

```
$ hv_open_example rt-linux
$ hv_sync_example rt-linux
$ cd /hv/guests/guestrtlinux
```

The following operations are supported to operate with guests:

- `hv_vmf_start [config_file]`: Load the VMF and the hypervisor configuration. If no configuration file is given, the one in the current guest folder is used.
- `hv_vmf_is_loaded`: Returns information if the VMF is loaded
- `hv_vmf_stop`: Stop all RTOS guests and unload VMF. It is **not** recommended to use this command while guests are running.
- `hv_guest_start [-view]`: Boot a currently powered off guest. The `-view` option will automatically open the console to interact with the guest.
- `hv_guest_restart [-view]`: Reboot a guest without reloading the Hypervisor configuration (only supported for RTOS guests).
- `hv_guest_console`: Open the console to interact with the guest. The console for RTOS guests is a shell like interface, the console for KVM guests typically is the guest desktop.
- `hv_guest_stop`: Stop the guest.

Caution: Only **one** debug console windows for each RTOS guest is supported. If multiple console windows are opened, the behavior is undefined.

5 RTOS Guests (RT-Linux, VxWorks, On Time RTOS-32, ...)

Important: You need to be familiar with chapter *Hypervisor Guests - General* before reading this chapter!

5.1 RTOS guests, general

Some ready to use RTOS images (containers) are provided, they are located in the respective template directory beyond `/hv/templates` (e.g. `/hv/templates/rt-linux` for RT Linux containers).

The first time an RTOS container is started, the RTOS Virtual Machine Framework (VMF) is loaded. Loading the VMF will also load the hypervisor configuration stored in the `hv.config` configuration file. This hypervisor configuration describes all guests and their related RTOS guest ID which is used to *attach* the guest to the VMF. In case a configuration entry has changed, all RTOS containers need to be stopped and the RTOS Virtual Machine to be reloaded. Configuration files are stored in `*.config` files.

5.1.1 RTOS Fileserver

The Hypervisor Host includes a file server which by default exposes the `/hv/guests` folder to RTOS guests.

The following configuration entry defines the root folder of the file server:

```
[Host\FileServer]
    "HomeDir"="/hv/guests"
```

The root folder is defined in `/hv/config/hvbase.config` for a System Manager based guest configuration or in one of the `*.config` files of the example guests.

This file server will use proprietary methods (using the Hypervisor's RTOS-Library) to expose files in this folder. The RT-Linux and RTOS-32 guests include a filesystem driver for these methods.

5.1.2 RTOS guest folder

Besides the standard files described in *General guest folder content*, all RTOS guest folders contain the following additional file(s).

- `device.config`: guest specific device configuration, you must **not** change this file

Additional configuration files may exist depending on the guest type and operation mode.

5.1.3 Log files

The following log files (located in the guest folder) will be created when starting a guest:

- `shutdown_svc.log` guest shutdown service log messages.
- `shutdown_hook.log` guest shutdown hook log messages.

5.2 Example Real-time Linux guest

See Real-time Linux Guest.

5.3 Example VxWorks guest

After the example is initialized, the VxWorks guest is located in `/hv/guests/guestvxworks`.

```
hv_open_example vxworks
hv_sync_example vxworks
cd /hv/guests/guestvxworks
```

Important: The shipped VxWorks image can only be executed if it is loaded into memory at a fixed address of `64` MByte. In the default configuration this is the case. When using the System Manager tool it must be configured as the very first Real-time guest.

5.3.1 File system access for VxWorks guests

To access the Hypervisor filesystem from within VxWorks, you may use FTP.

Follow these instructions to install an FTP server in the Hypervisor in case it is not yet installed.

- activate the entry `write_enable=YES` in `/etc/vsftpd.conf` (you need to edit it with `root` rights)
- create a new entry at the and of this file (this will become the root of the ftp filesystem): `local_root=/hv/guests`
- Restart the FTP server: `sudo systemctl restart vsftpd`
- **Create an FTP user with the appropriate VxWorks client username and password (default is `target` with password `vxworks`):**
  - run `sudo adduser target`
  - use the following password: `vxworks`
- After executing the above steps, from within VxWorks, you should be able to access the guest folder via the VxWorks path `pc:`
- **If you need write access for the FTP client, you should do the following:**

- add the user `target` to the current user group: `sudo usermod -aG $(whoami) target`
- for each folder that shall get write access run `sudo chmod 775 /path/to/folder`

5.4 Example On Time RTOS-32 guest

After the example is initialized, the RTOS-32 guest is located in `/hv/guests/guestrtos32`

```
hv_open_example rtos-32
hv_sync_example rtos-32
cd /hv/guests/guestrtos32
```

The guest image is selected through the `osImage` variable in the configuration script.

By default it is the RTOS-32 loader image `/hv/templates/rtos-32/Loader.bin` which is set in `/hv/guests/guestrtos32/guest_config.sh`. If you want to change the RTOS-32 image, you should set the `osImage` variable in the user specific guest configuration script `usr_guest_config.sh`.

If the `Loader.bin` is used, the loader will use the application DLL which is defined in the following configuration section in `/hv/guests/guestrtos32/guest.config`:

```
[Rtos\Loader]                           ; Used by Loader.bin
     "DllName"="rtos32app.dll"          ; File or link must exist in the␣
↪guest directory
```

By default this is a link to the effective DLL: `/hv/guests/guestrtos32/rtos32app.dlm` which by default points to the `RTOS-32Demo.dlm` demo application.

You may select a different application, for example the `RealtimeDemo` as follows:

```
$ cd /hv/guests/guestrtos32
$ rm rtos32app.dlm
$ ln -s files/RealtimeDemo.dlm rtos32app.dlm
```

5.4.1 File system access for RTOS-32 applications

From within the RTOS-32 application you may access the Hypervisor filesystem via the `C:` drive. The root directory is set in `guest.config` in section `[\\Host\\FileServer]`. By default, it is set to the example guest folder at `/hv/guests/guestrtos`.

5.5 RTOS Shared Mode operation

> **Caution:** This feature is **not** supported in RTOSVisor V8.1!

Using the RTOSVisor, Intel(c) VT technology allows to activate so called “Shared Mode”. In Shared Mode it is possible to run an RTOS on the same CPU as the Hypervisor Host (thus, the CPU is shared between two OSes). Please note, that hard real time for RTOS is also achieved in this mode.

Intel(c) VT-x and VT-d are hardware virtualisation extensions for Intel Processors. Please take a look at the official Intel page for full description of the technology: https://www.intel.com/content/www/us/en/virtualization/virtualization-technology/intel-virtualization-technology.html

5.5.1 Activating VT

VT should be activated *manually*. In this example we configure a `2` CPU system, where the first CPU is used to run Hypervisor Host and RTOS1 (so, in shared CPU mode) and the second CPU will be used to run RTOS2 (as exclusive core).

This script installs necessary `.deb` packages, configures memory settings, reads `ACPI` tables and etc.

5.5.2 Linux Kernel grub parameters

IOMMU must be *disabled* in order to activate VT-D.

Add the following parameters to the linux kernel in `grub.cfg`: `intremap=off intel_iommu=off maxcpus=1`

Make sure these parameters have been removed: `iommu=pt vfio_iommu_type1.allow_unsafe_interrupts=1`

Currently **only** one shared cpu is supported. This is the reason, why `maxcpus=1` is used.

5.5.3 Adapt `.config` files

1. **`[Upload]`**
   `"VersionDrv"=dword:9070000`
2. **`[Vmf]`**
   `"VtAllowed"=dword:1`
3. **`[Rtos]`**
   `"MemoryType"=dword:3`
4. **`[Rtos1]`**
   `"MemoryType"=dword:3`
5. **`[Rtos\Vmf]`**
   `"MapSystemTables"=dword:1`

5.6 Achieving Hard Real-Time Capabilities

5.6.1 Background information

It is important to understand, which factors have influence to the real-time capabilities of your hardware. Most important of them are BIOS Settings (no CPU power saving modes should be activated, no CPU throttling, no variable CPU Frequency, USB Legacy support should also be disabled). System Management Interrupts (SMIs) must also be avoided.

Graphics

The video card has usually a huge impact on the real-time capabilities of the real-time system, because it cannot only generate SMI but also perform huge background DMA transfers from/to DRAM memory.

For example, if you have the Intel i915 video card on your Hypervisor Host, its Linux driver can produce significant interrupt/context latencies.

As a first step, in order to achieve better latencies, disable the LightDM graphics manager in your RTOSVisor host operating system. Simply run the following command `sudo apt-get remove lightdm` and then reboot your system.

This step converts your system into console only mode, no GUI. But it is not enough, because a video card can do DMA Transfers even without GUI.

The easiest and fastest solution here is to kill its driver as follows:

- establish a connection with Hypervisor Host via SSH (port 22)
- kill the video driver `sudo rmmod -f i915`. Now the computer monitor **cannot** be used anymore.
- start realtime demo and make sure, that context switch/interrupt delays much better now (200%+):

CPU Core frequency policy

Intel processors with Hardware P-State (HWP) support delegate per-core P-state control to on-chip firmware via MSRs (Model-Specific Registers), rather than relying on legacy ACPI governors. HWP-capable CPUs expose:

- **IA32_HWP_CAPABILITIES (MSR 0x771)** Describes the processor's guaranteed performance range (min/max HWP indices), supported energy-performance preferences (EPP), and platform limits.
- **IA32_HWP_REQUEST (MSR 0x774)** Allows software to request a target performance index and EPP hint on a per-core basis.

When HWP is enabled, the hardware governor automatically adjusts core frequency and voltage to satisfy requests written into IA32_HWP_REQUEST, providing smooth transitions and power-efficient operation. This can be used to optimize the performance and deterministic behavior for CPU cores used by Real-time operating systems.

Two commands simplify inspecting and updating per-core HWP settings:

- **hv_get_cpu_performance <CPU_ID>**
  1. Verifies the logical CPU is online.
  2. Reads MSR 0x771 to obtain the "highest performance" index.
  3. Reads MSR 0x774 and decodes its four 8-bit fields: Desired, Minimum, Maximum, and EPP.
  4. Prints each field (hex and decimal) plus "Desired/Highest (%)" for quick comparison.
- **hv_set_cpu_performance <CPU_ID> <PERCENTAGE>**
  1. Aborts if the CPU is offline ("CPU <CPU_ID> is offline.").
  2. Reads MSR 0x771 to determine the maximum HWP index.
  3. Calculates:

```
target = max(1, ceil(max_index × PERCENTAGE / 100))
```

ensuring any non-zero percentage maps to at least index 1.

4. Constructs a 32-bit HWP_REQUEST (Desired = Min = Max = target, EPP = 0x00 for maximum performance) and writes it to MSR 0x774.
5. Prints the hex value written and a confirmation message.

Once a RTOS was started, the CPU may be offline. If a logical CPU is offline, the above commands will not work.

To bring a core back online:

```
hv_set_cpu_online <CPU_ID>
```

This script checks VMF status and, if unloaded, writes "1" to `/sys/devices/system/cpu/cpu<CPU_ID>/online`. Note that CPU 0 has no "online" file (always online), and the command will report that accordingly.

5.6.2 Real-time measurement

RT-Linux realtime demo

The shipped RT-Linux guests include a real-time measurement application.

Follow these steps to initialize the RT-Linux example and run the measurement:

```
$ hv_open_example rt-linux
$ hv_sync_example rt-linux
$ cd /hv/guests/guestrtlinux
$ hv_vmf_stop
$ hv_guest_start -view
```

Then log in (user = root, password = root)

```
$ RealtimeDemo
```

RTOS-32 realtime demo

You can use the RTOS-32 realtime demo application to determine the real-time capabilities of your system.

If you did not change the shipped RTOS-32 configuration, follow these steps to initialize the RTOS-32 example and run the measurement:

Initialize the RTOS-32 example

```
$ hv_open_example rtos-32
$ hv_sync_example rtos-32
$ cd /hv/guests/guestrtos32
$ mousepad usr.config
```

Add the following lines to modify the Fileserver path directing to the guest directory.

```
[Host\FileServer]
    "HomeDir"="/hv/guests/guestrtos32"
```

Update the link and start the application:

```
$ rm -f rtos32app.dlm
$ ln -s /hv/templates/example_guests/rtos-32/files files
$ ln -s /hv/guests/guestrtos32/files/RealtimeDemo.dlm rtos32app.dlm
$ hv_guest_start -view
```

6 KVM Guests (Windows, Ubuntu, Debian, ...)

Important: You need to be familiar with chapter *Hypervisor Guests - General* before reading this chapter!

6.1 KVM guests, general

There are no example KVM guests shipped with the RTOSVisor. Instead, these operating systems have to be installed under control of the hypervisor from an installation media (an ISO file).

6.2 Filesystem access and file sharing

6.2.1 Default SMB share

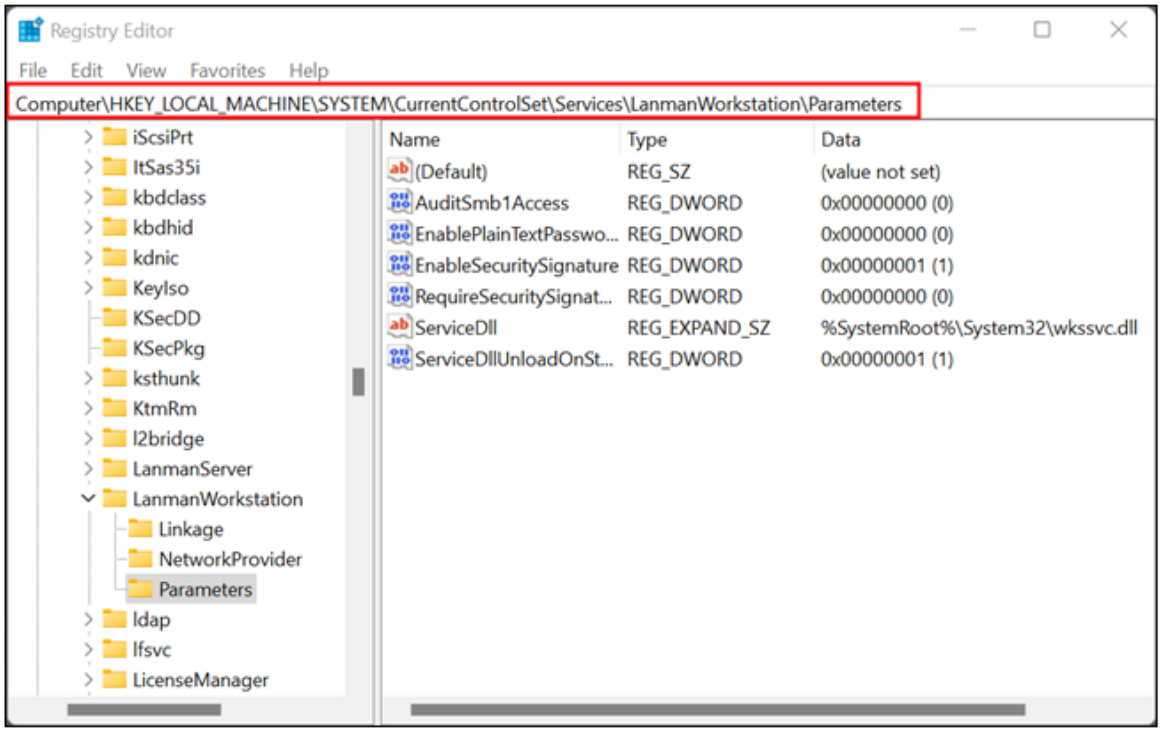
The Hypervisor Host by default exposes the `/hv/guests` folder to KVM guests via a SMB share (Windows file share). The share can be accessed from within the guest using the IP address `10.0.2.4` and the share name `qemu`. See more here: *Default SMB share*.

Using the default SMB share on Windows guests

On Windows the default SMB share can be accessed through `\\10.0.2.4\qemu`.

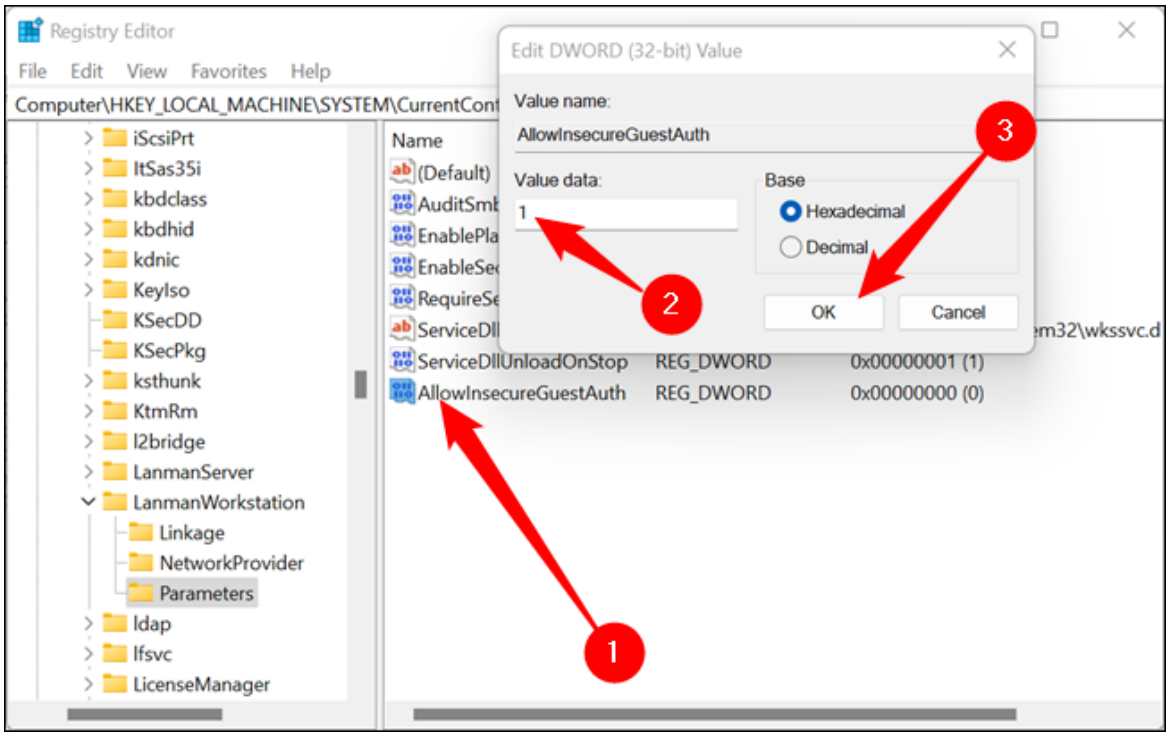
Caution: You may get an error (0x80004005) when accessing the share due to some restrictive Windows settings. In that case, try to allow insecure guest logins. Windows blocks guest logins to network devices using SMB2 by default. You might need to disable that setting.

Open the Registry Editor and then navigate to HKLM\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parame
using the menu on the left, or just paste the path into the address bar.



The `DWORD` you're looking for is named `AllowInsecureGuestAuth` — if it isn't there, you'll need to create it.

Right-click empty space, mouse to "New," then click "DWORD (32-bit) Value." Name it `AllowInsecureGuestAuth` and set the value to `1`.



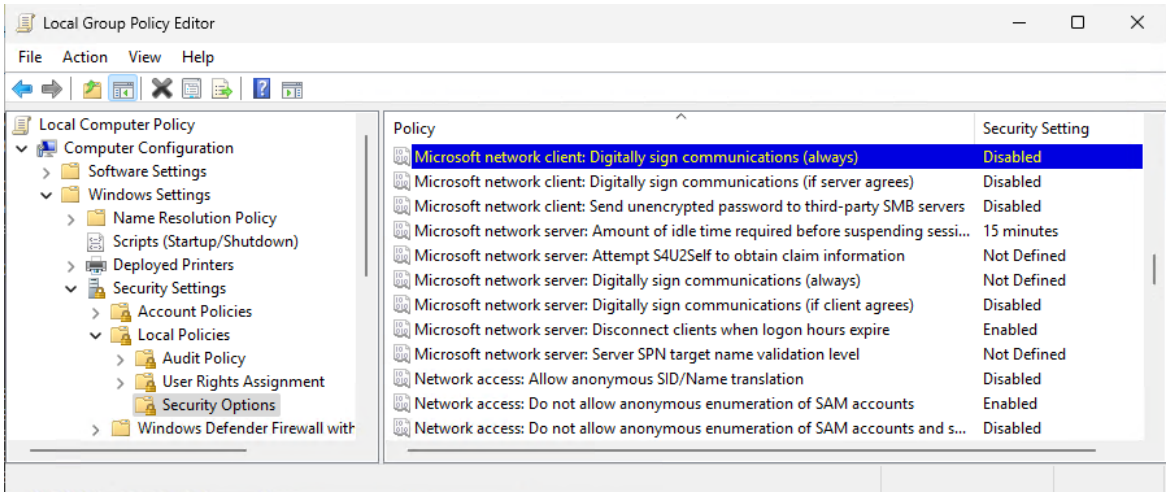
Caution: Another issue that could block access to the network share is SMB signing. You may encounter a message like *You can't access this shared folder because your computer is configured to require SMB signing.* Then, please disable signing via the appropriate policies.

Open the policy editor gpedit.msc.

Navigate to Computer Configuration - Windows Settings - Security Settings - Local Policies - Security Options.

Locate policies such as *Microsoft network client: Digitally sign communications (always)* and *Microsoft network server: Digitally sign communications (always)*.

Disable these policies



Using the default SMB share on Ubuntu/Debian guests

To mount the default SMB share the network client software (`cifs-utils`) must be installed and then the SMB share can be mounted. In the below example, we will mount the share at `/mnt/qemu`:

```
$ sudo apt-get update
$ sudo apt-get install cifs-utils
$ sudo mkdir /mnt/qemu
$ sudo mount -t cifs //10.0.2.4/qemu /mnt/qemu -o guest
```

6.2.2 SMB (Windows) file share

Instead of using the (limited) default SMB share, you may use a full featured SMB solution using SAMBA. On the hypervisor, SAMBA must be installed in a first step: *SMB (Windows) file share*

Using the SMB share on Windows guests

Using the command

```
$ net view
192.168.178.156
```

you can see which SMB shares are available on the hypervisor (assuming its IP address is 192.168.178.156).

Alternatively, try to access the SMB share from the Windows explorer (we assume, the share name is guests). You may have to use the IP address of the hypervisor.

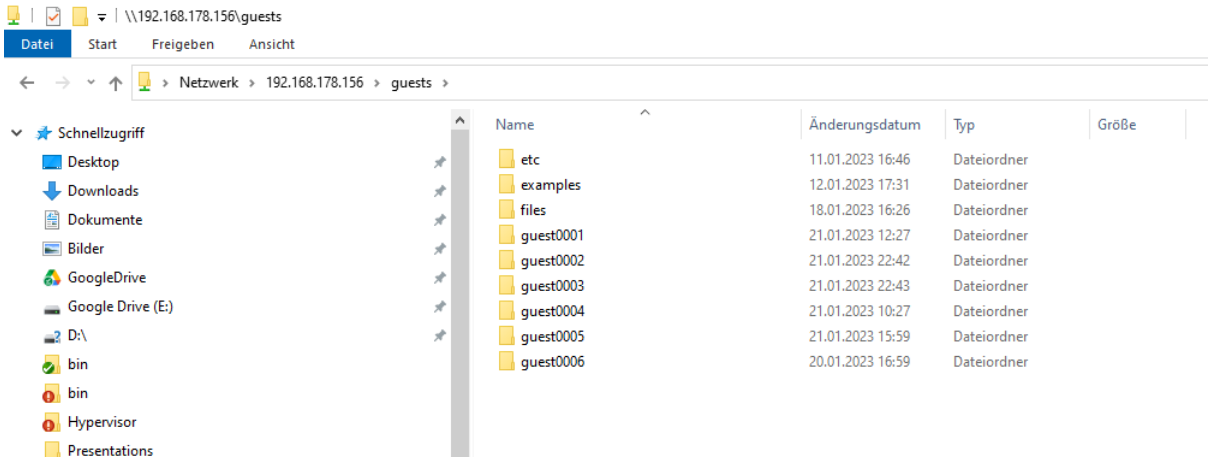


Fig. 6.1: Access to share from Windows file explorer.

Using the SMB share on Ubuntu/Debian guests

To mount the SAMBA share the network client software (cifs-utils) must be installed and then the SMB share can be mounted. In the below example, we will install the required packages and check which shares are available:

```
$ sudo apt-get update
$ sudo apt-get install cifs-utils
$ sudo apt-get install smbclient
$ smbclient -L 192.168.178.156 -N
```

Then we will mount the guests share at /mnt/guests, the username should be valid on both, the guest and the Hypervisor host:

```
$ sudo mkdir /mnt/guests
$ sudo mount -t cifs -o username=hvuser //192.168.178.156/guests /mnt/guests
```

To automatically mount the SMB share on boot, add an entry to the /etc/fstab file on the client machine:

1. Edit the fstab File

```
$ sudo nano /etc/fstab
```

2. Add the SMB Share

```
$ sudo mousepad /etc/fstab
$ //192.168.178.156/guests /mnt/guests cifs
↪  credentials=/etc/samba/credentials,uid=hvuser,gid=hvuser,iocharset=utf8
↪  0 0
```

3. Save and Exit
4. Secure Credentials

For security reasons, we are using credentials stored in a separate file and refer to it from /etc/fstab.

```
$ sudo mousepad /etc/samba/credentials
```

Add the following content:

```
username=smbuser
password=smbpass
```

Secure the credentials file:

```
$ sudo mkdir /etc/samba
$ sudo chmod 600 /etc/samba/credentials
```

5. Test the fstab Entry

```
$ sudo mount -a
```

6.2.3 NFS share

An alternative to SMB is the Network Filesystem (NFS). The NFS server must be installed in a first step: *NFS access*

Using the NFS share on Windows guests

Step 1: Install NFS Client Open Control Panel:

- Press Win + X and select “Apps and Features”.
- Click on “Programs and Features” on the right.
- Select “Turn Windows features on or off”.

Enable NFS Client:

- In the Windows Features dialog, scroll down and check “Services for NFS”.
- Click “OK” and wait for the installation to complete.
- Restart your computer if prompted.

Step 2: Mount the NFS Drive Open Command Prompt:

Press Win + X and select “Command Prompt (Admin)” or “Windows PowerShell (Admin)”.

Mount the NFS Share:

We can check which shares are exposed:

```
$ showmount -e <NFS Server>
```

Use the mount command to mount the NFS share. The syntax is:

```
$ mount -o anon

<NFS Server>
<NFS Share> <Drive Letter>:
```

Note: Ensure that you replace `<NFS Server>` with the IP address or hostname of your NFS server, `<NFS Share>` with the name of the NFS share, and `<Drive Letter>` with the desired drive letter (e.g., `X:`).

For example:

```
$ mount -o anon

192.168.1.100
shared_folder X:
```

Or alternatively:

```
$ net use X:

192.168.1.100
shared_folder X: /user:username password
```

Using the NFS share on Ubuntu/Debian guests

To mount a NFS share, the NFS client has to be installed. We will also check which shares are exposed. In case you have uninstalled the NFS client, re-install the NFS client again:

```
$ sudo apt update
$ sudo apt install nfs-common
$ showmount -e 192.168.1.100
```

Then a mount point must be created. We assume the hypervisor exposes the `/hv/guests` folder.

```
$ sudo mkdir -p /mnt/hv
$ sudo mkdir -p /mnt/hv/guests
```

Then we will mount the NFS Share:

```
$ sudo mount -v -o vers=4,nolock,proto=tcp 192.168.1.100:/hv/guests
↪  /mnt/hv/guests
```

Replace `192.168.1.100` with the IP address of your NFS server.

To automatically mount the NFS share on boot, add an entry to the `/etc/fstab` file on the client machine:

1. Edit the `fstab` File

```
$ sudo nano /etc/fstab
```

2. Add the NFS Share

```
$ sudo mousepad /etc/fstab
$ 192.168.1.100:/hv/guests /mnt/hv/guests nfs defaults 0 0
```

3. Save and Exit
4. Test the fstab Entry

```
$ sudo mount -a
```

6.3 Communication Subsystem

A KVM guest may use the communication subsystem which provides specific functions to communication with other guests:

- Direct access to the Virtual Network (compared to bridged access if the communication subsystem is not used)
- RTOS-Library functions (Shared Memory, Pipes, Events etc.)

Hint: The communication subsystem is part of the RTOS Virtual Machine Framework (VMF)

To enable access to the communication subsystem, the following settings are required in the `guest_config.sh` or `usr_guest_config.sh` configuration file:

```
export rtosOsId=#####
export hvConfig=%%%%%
```

The `rtosOsId` value needs to be a unique RTOS guest id, this id is used by the guest to attach to the communication subsystem. Valid ids are in the range from `1` to `4`. The `hvConfig` value will have to point to the appropriate `hv.config` file used for the hypervisor configuration.

The very first time, when an RTOS container or such a KVM guest is started, the RTOS Virtual Machine Framework (VMF) is loaded. Loading the VMF will also load the hypervisor configuration stored in the `hv.config` configuration file. This hypervisor configuration describes all guests and their related guest ID which is used to *attach* the guest to the VMF. In case a configuration entry has changed, all RTOS containers need to be stopped and the RTOS Virtual Machine to be reloaded. Configuration files are stored in `*.config` files.

6.3.1 KVM guest folder

Besides the standard files described in *General guest folder content*, all KVM guest folders contain the following additional file(s).

- `OVMF_CODE.fd`: UEFI firmware image
- `OVMF_VARS.fd`: UEFI firmware variable store
- `guest_gateway.config`: Configuration for guest port forwarding from an external ethernet network to the virtual network

Additional configuration files may exist depending on the guest type and operation mode.

Hint:

By specifying the `Disk Image File` in the system manager project, the location and name of the guest image files can be determined. The `Disk Image File` entry always consists of a path (starting with `/hv/VMs`) and filename with the `.qcow2` extension (e.g., `/hv/VMs/windows/win10_20h2.qcow2`).
The UEFI firmware files will also be stored in this path.

6.4 KVM Guest Operation

Besides the commands described in *Guest operation* the following KVM guest specific commands are available.

- `hv_guest_start [-view | -kiosk]`: Start the guest. The `-view option` will show the guest desktop in standard mode (resizable window), the `-kiosk` option in full screen mode.
- `hv_guest_console [-kiosk]`: Show guest desktop in full screen mode (`-kiosk` option).
- `hv_guest_stop [-reset | -kill]`: Reset (`-reset` option) or Power off (`-kill` option) the guest (instead of gracefully shutdown without any option).
- `hv_guest_monitor`: Start the KVM guest monitor. Monitor commands are described in here: https://en.wikibooks.org/wiki/QEMU/Monitor

6.4.1 Kiosk mode

If you want to generally enable the kiosk mode after the next start of the guest VM, adjust the guest configuration:

```
$ cd GUEST_FOLDER
$ mousepad usr_guest_config.sh
```

```
 export kiosk_mode=1
```

6.4.2 Displaying the guest desktop

To display the guest desktop, you need to run the `hv_guest_console` command from within the guest folder. Typically you need to first log in and then start the guest and the console. It will then be displayed on the RTOSVisor desktop.

Alternatively it is also possible to use X11 forwarding in an SSH session. The display device to be used is set in the `DISPLAY` environment variable. A local display is identified by `:0.0`, an X11 forwarded display can look like `localhost:10.0`. You can determine it as follows:

```
$ echo $DISPLAY
```

To check if the display is accessible, run:

```
$ [[ $(xset -q 2 2>/dev/null) ]] && echo "display works"
```

If the result is `display works`, then everything is set correctly.

Hint: The display is accessed via the X protocol which needs authentication. The authentication file is defined in the `XAUTHORITY` environment variable. Typically it is located in the user's home folder: `/home/MyUser/.Xauthority`. If for some reason this file does **not** exist, you may **turn off** authentication:

```
$ xhost +
```

NOTE: The `root` user typically does not have access to the display.

Hint: This section applies to the System Manager. By default, the System Manager is automatically started as a service without any display access. Thus, the desktop (guest console) for KVM guests cannot

be launched by default. To enable launching the console from within the System Manager, you may have to log in and restart the System Manager service:

```
$ hv_sysmgr restart
```

You typically restart the service from within the RTOSVisor locally (by logging in in its desktop), in that case the RTOSVisor desktop will be used to show the guest desktop. Alternatively, if you restart the service inside an SSH session, there will be two options:

- If X11 forwarding is enabled, the guest desktop will be forwarded to a remote display.
- If X11 forwarding is disabled, the local desktop will be used. In this case you need to login first before launching the console.

6.4.3 Log files

The following log files (located in the guest folder) will be created when starting a guest:

- `qemuif.log` log messages of setting up network bridging on guest boot and terminate network bridging on guest shutdown.
- `kvmguest.log` guest VM logging (KVM hypervisor).
- `kvmview.log` guest VM viewer log file.
- `shutdown_svc.log` guest shutdown service log messages.
- `shutdown_hook.log` guest shutdown hook log messages.

6.5 Example KVM Windows guest

The example KVM Windows guest configuration is located in `/hv/guests/guestwindows` after initializing the example. You need to install the Windows guest using an ISO installation media. Please read the `Windows Guest Guide` for more information about Windows guests.

Hint: Initially, the Hypervisor Host does not provide any example guest folders. To switch to an example guest, you must execute the corresponding initialization. For instructions on how to initialize the examples, refer to *Example guest folders*.

6.6 Example KVM Ubuntu guest

The example KVM Ubuntu guest configuration is located in `/hv/guests/ubuntu` after initializing the example. You need to install the Ubuntu guest using an `ISO` installation media. Please read the `Ubuntu Guest Guide` for more information about Ubuntu guests.

Hint: Initially, the Hypervisor Host does not provide any example guest folders. To switch to an example guest, you must execute the corresponding initialization. For instructions on how to initialize the examples, refer to *Example guest folders*.

6.7 KVM guest basic settings

Hint:

The term `GUEST_FOLDER` is related to the folder where the guest configuration files are located (e.g. `/hv/guests/guestwindows`).
The term `GUEST_NAME` is related to the guest name (e.g. `windows`).

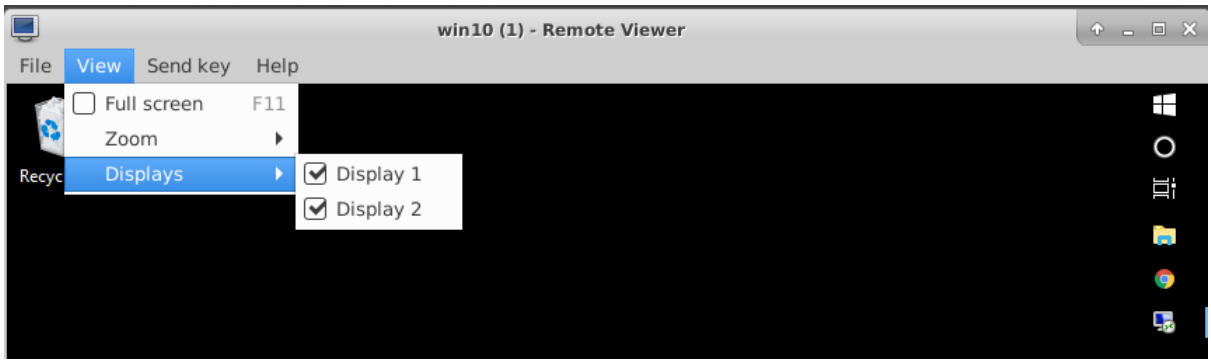
6.7.1 Guest multiple monitors

If more than one monitor is connected to the system, these can also be used for VM guests (up to a maximum of `4` monitors).

For Windows guests, the following settings are required in `usr_guest_config.sh`:

```
export num_monitors=# (where # is the number of monitors)
```

To enable additional monitors being displayed, select the displays via the View – Displays menu in the viewer application.



6.7.2 Guest Multi Touch

In KVM Windows guests, multitouch functionality is available. Therefore, it is necessary to display the guest in full-screen mode either by using remote-viewer or kiosk mode.

To enable multitouch, it's necessary to identify the corresponding touch event on the Hypervisor Host:

```
ls -la /dev/input/by-id | grep -event-
```

The following settings are required in `usr_guest_config.sh`:

```
export enable_multitouch=1
export multitouch_event_device=""  # fill in the corresponing touch device
```

In the following example you would use `usb-wch.cn_TouchScreen_9LQ0172005164-event-if00` as `multitouch_event_device`.

```
hvuser@PC-035476:~$ ls -la /dev/input/by-id | grep -event-
lrwxrwxrwx 1 root root   9 Jul  2 12:36 usb-LITEON_Technology_Corp._HP_125_
↪Wired_Keyboard-event-if01 -> ../event6
lrwxrwxrwx 1 root root   9 Jul  2 12:36 usb-LITEON_Technology_Corp._HP_125_
↪Wired_Keyboard-event-kbd -> ../event5
lrwxrwxrwx 1 root root   9 Jul  2 12:36 usb-PixArt_HP_125_USB_Optical_
↪Mouse-event-mouse -> ../event8
lrwxrwxrwx 1 root root   9 Jul  2 12:36 usb-wch.cn_TouchScreen_
↪9LQ0172005164-event-if00 -> ../event3
lrwxrwxrwx 1 root root   9 Jul  2 12:36 usb-wch.cn_TouchScreen_
↪9LQ0172005164-if02-event-mouse -> ../event2
```

6.7.3 QXL - Graphics Memory

Graphics memory is the dedicated RAM on the virtual GPU that holds framebuffers, textures and other graphical data used by the guest OS. In RTOSVisor QEMU/KVM, the QXL driver is chosen for Windows guests without multitouch support, which provides efficient 2D/3D acceleration. You can control how much graphics memory is allocated by defining `vga_memsize_mb` passed to the QXL VGA device.

By default you can leave it at 64 MiB for a single, low-resolution display, but if you're driving two or more monitors (or anything above 1920×1080) you'll want at least 128 MiB so the guest has enough video RAM to hold all your desktop surfaces.

The settings can be adjusted in `usr_guest_config.sh`:

```
export vga_memsize_mb=64
```

Hint: If you encounter video-memory allocation errors in your dmesg or Xorg logs, try increasing the `vga_memsize_mb` setting to allocate more VRAM to the guest.

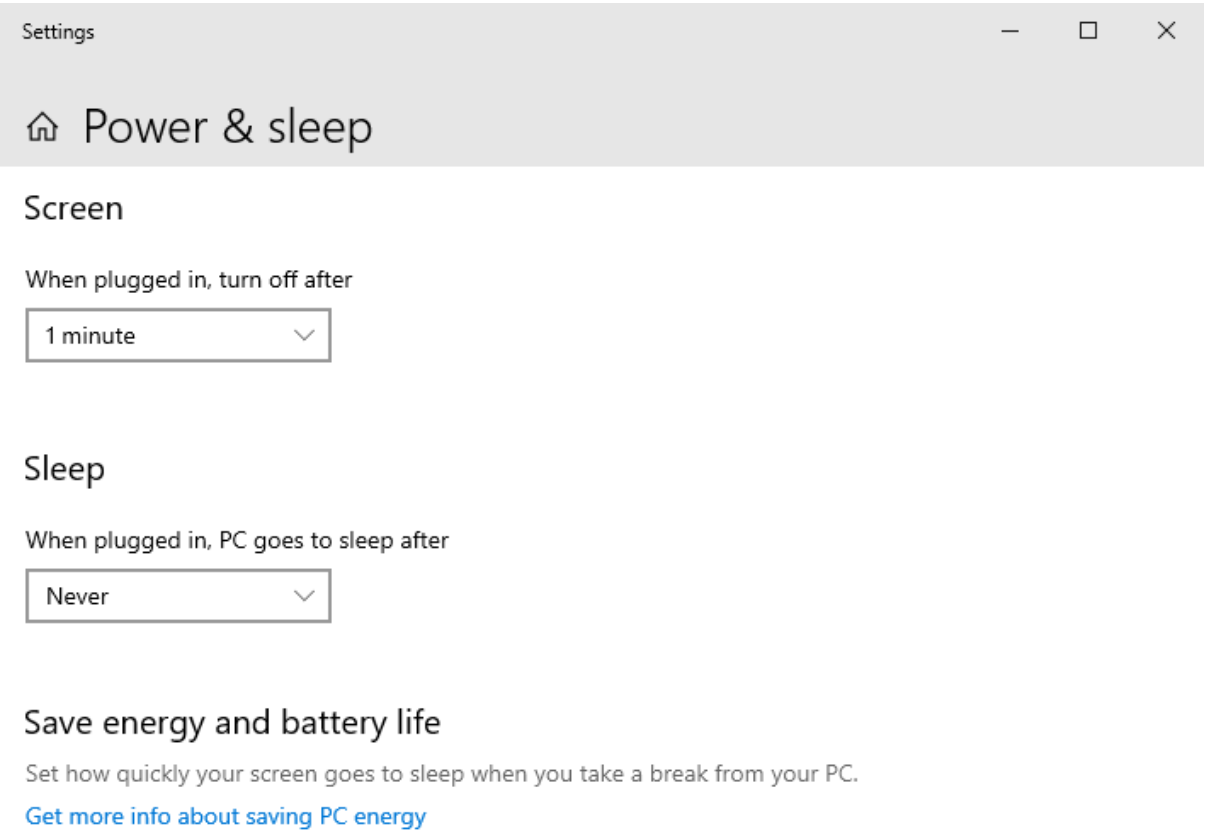
```
[   38.035369] qxl 0000:00:02.0: object_init failed for (9220096,␣
↪0x00000001)
[   38.035389] [drm:qxl_alloc_bo_reserved [qxl]] *ERROR* failed to␣
↪allocate VRAM BO
[   58.259390] qxl 0000:00:02.0: object_init failed for (9220096,␣
↪0x00000001)
```

6.8 Windows installation

Attention: See **Hypervisor - Windows Guest Guide** for tutorial.

6.8.1 Additional settings for Windows guests:

Disable Sleep (*Windows Settings -> System -> Power and sleep*):



6.9 Ubuntu installation

Attention: See **Hypervisor - Ubuntu Guest Guide** for tutorial.

6.10 Windows/Linux COM port guest access

It is possible to passthrough a `COM` port to a guest VM. It works for both legacy serial ports and USB-to-Serial converters.

Modify the `usr_guest_config.sh` file and change

```
export OTHER_HW=$OTHER_HW
```

to

```
export OTHER_HW=$OTHER_HW"  -serial /dev/ttyUSB0 -serial /dev/ttyS0"
```

This example creates `COM1` port in the VM and it uses USB-to-Serial converter on a Hypervisor Host and creates `COM2` port in the VM and it is the real `COM1` port on Hypervisor Host.

6.11 PCI Device passthrough (Windows/Linux)

6.11.1 Why pass-through

Typically, Windows or Linux guest operating systems will run in a sandbox like virtual machine with no direct hardware access. There are scenarios when this is not sufficient, for example some PCI devices (e.g. CAN cards) will not be virtualized and thus would not be visible in such a guest OS. There may also be significant performance impacts in some cases if virtual hardware is used (especially for graphics hardware). To overcome these limitations, the guest will have to use the real physical hardware instead of virtual hardware. PCI Device passthrough will directly assign a specific PCI device to a Windows or Linux guest.

It is **mandatory** to have hardware support for IOMMU (VT-d). VT-d should be supported by your processor, your motherboard and should be enabled in the BIOS.

Virtual Function I/O (VFIO) allows a virtual machine to access a PCI device, such as a GPU or network card, directly and achieve close to bare metal performance.

The setup used for this guide is:

- Intel Core I5-8400 or I3-7100 (with integrated Intel UHD 630 Graphics) - this integrated graphics adapter will be assigned to the Windows VM.
- AMD/ATI RV610 (Radeon HD 2400 PRO) – optional, as a second GPU, only needed to have display output for Hypervisor Host. Later, the Hypervisor Host is reached only via SSH.
- Intel I210 Gigabit Network Card - optional, to demonstrate how to pass through a simple PCI device to a Windows VM.

6.11.2 Ethernet PCI Card/Custom PCI device assignment

Some manual work is required to pass through the PCI Device to a Windows VM.

Understanding IOMMU Groups

In order to activate the hardware passthrough we have to prevent the ownership of a PCI device by its native driver and assign it to the `vfio-pci` driver instead.

In a first step, an overview of the hardware and related drivers is required. In the below example we want to passthrough the I210 Ethernet Controller to the guest VM.

```
rte@rte-System-Product-Name:~$ lspci
00:00.0 Host bridge: Intel Corporation 8th Gen Core Processor Host Bridge/
↪DRAM Registers
00:01.0 PCI bridge: Intel Corporation Xeon E3-1200 v5/E3-1500 v5/6th Gen␣
↪Core Processor
00:02.0 VGA compatible controller: Intel Corporation Device 3e92
00:14.0 USB controller: Intel Corporation Cannon Lake PCH USB 3.1 xHCI␣
↪Host Controller
00:14.2 RAM memory: Intel Corporation Cannon Lake PCH Shared SRAM (rev 10)
00:16.0 Communication controller: Intel Corporation Cannon Lake PCH HECI␣
↪Controller
00:17.0 SATA controller: Intel Corporation Cannon Lake PCH SATA AHCI␣
↪Controller (rev 10)
```

(continues on next page)

(continued from previous page)

```
00:1c.0 PCI bridge: Intel Corporation Device a33c (rev f0)
00:1c.5 PCI bridge: Intel Corporation Device a33d (rev f0)
00:1c.7 PCI bridge: Intel Corporation Device a33f (rev f0)
00:1f.0 ISA bridge: Intel Corporation Device a303 (rev 10)
00:1f.4 SMBus: Intel Corporation Cannon Lake PCH SMBus Controller (rev 10)
00:1f.5 Serial bus controller [0c80]: Intel Corporation Cannon Lake PCH␣
↪SPI Controller
01:00.0 VGA compatible controller: [AMD/ATI] RV610 [Radeon HD 2400 PRO]
01:00.1 Audio device: [AMD/ATI] RV610 HDMI Audio [Radeon HD 2400 PRO]
03:00.0 Ethernet controller: Intel Corporation I210 Gigabit Network␣
↪Connection (rev 03)
04:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111/8168/
↪8411 PCI Express Gigabit Ethernet Controller (rev 15)
```

The I210 device is determined as `03:00.0`

Caution: The IOMMU has to be activated, you can verify this as follows:

```
sudo dmesg | grep -e "Directed I/O"
DMAR: Intel(R) Virtualization Technology for Directed I/O
```

Hint: In case the IOMMU is activated, all devices are divided into groups. The IOMMU Group is an indivisible unit. All devices in the same group must be passed through together, it is not possible to only pass through a subset of the devices.

In the next step, we need to get an overview of the IOMMU architecture and determine to which group the device we want to pass through belongs to.

```
for a in /sys/kernel/iommu_groups*; do find $a -type l; done | sort --
↪version-sort
```

```
/sys/kernel/iommu_groups/0/devices/0000:00:00.0
/sys/kernel/iommu_groups/1/devices/0000:00:01.0
/sys/kernel/iommu_groups/1/devices/0000:01:00.0
/sys/kernel/iommu_groups/1/devices/0000:01:00.1
/sys/kernel/iommu_groups/2/devices/0000:00:02.0
/sys/kernel/iommu_groups/3/devices/0000:00:14.0
/sys/kernel/iommu_groups/3/devices/0000:00:14.2
/sys/kernel/iommu_groups/4/devices/0000:00:16.0
/sys/kernel/iommu_groups/5/devices/0000:00:17.0
/sys/kernel/iommu_groups/6/devices/0000:00:1c.0
/sys/kernel/iommu_groups/7/devices/0000:00:1c.5
/sys/kernel/iommu_groups/8/devices/0000:00:1c.7
/sys/kernel/iommu_groups/9/devices/0000:00:1f.0
/sys/kernel/iommu_groups/9/devices/0000:00:1f.4
/sys/kernel/iommu_groups/9/devices/0000:00:1f.5
/sys/kernel/iommu_groups/10/devices/0000:03:00.0
/sys/kernel/iommu_groups/11/devices/0000:04:00.0
```

The I210 device (`03:00.0`) belongs to the IOMMU group `10`. It is important to know that all devices in a single group are shared. No other device belongs to this IOMMU group so we can pass through the I210

device to the guest.

We need to determine the PCI vendor and device ID of the I210 device now:

```
rte@rte-System-Product-Name:~$ lspci -s 03:00.0 -vvn
03:00.0 0200: 8086:1533 (rev 03)
    ...
    Kernel driver in use: igb
    Kernel modules: igb
```

Now we can check whether multiple devices share the same `vendor:device` ID. In this example, we're checking for multiple Intel I210 devices. The output might look like this:

```
rte@rte-System-Product-Name:~$  lspci -nnk | grep 8086:1533
    02:00.0 Ethernet controller [0200]: Intel Corporation I210 Gigabit␣
↪Network Connection [8086:1533] (rev 03)
    03:00.0 Ethernet controller [0200]: Intel Corporation I210 Gigabit␣
↪Network Connection [8086:1533] (rev 03)
```

If we have found only one device or want to pass all identical devices to the KVM guest, you can continue with the chapter *Passing Devices Using Vendor:Device ID*.

If we have found multiple devices and want to pass only a single device (e.g., `03:00.0` - I210), proceed with the chapter *Passing Device by PCI Address*.

Hint: As mentioned before, device passthrough requires IOMMU support by the hardware and the OS. It is needed to add `intel_iommu=on iommu=pt` to your kernel command line. These parameters are added by RTOSVisor during the installation (via the `/hv/bin/inithv.sh` script which executed automatically by the system).

Typically, one or multiple PCI devices will also be assigned to an RTOS, for example Real-time Linux. In such a case (one or more PCI devices are passed through to a Windows or Ubuntu guest as well as one or multiple PCI devices are assigned to an RTOS) it is required to deactivate IOMMU Interrupt Remapping in the Linux Kernel.

The following kernel command line parameters usually are added as well by the `/hv/bin/inithv.sh` script into the GRUB entry "Hypervisor" in the `/etc/grub.d/40_custom` file and into the GRUB entry "Hypervisor + iGVT-d" in the `/etc/grub.d/41_custom` file. You may verify and add these parameters if they are missing. In that case it is also required to execute `update-grub` and then reboot.

```
intremap=off vfio_iommu_type1.allow_unsafe_interrupts=1
```

Passing Devices Using Vendor:Device ID

The following steps are required if we have found only one device, or if we want to pass all identical devices (e.g., all I210 devices) to the KVM guest.

To ensure that the `vfio-pci` driver is loaded before the kernel driver (in this case, `igb`) and automatically binds to the specified device, create or edit the following configuration file:

```
sudo mousepad /etc/modprobe.d/vfio.conf
```

Add the following lines (replace `igb` with the driver for your device, and `8086:1533` with the correct vendor and device ID for your hardware):

```
softdep igb pre: vfio-pci
options vfio-pci ids=8086:1533
```

After editing the configuration, regenerate the initramfs so the changes take effect on the next boot:

```
sudo update-initramfs -u
```

Add the following parameter to the Linux kernel command line: `vfio-pci.ids=8086:1533`. To modify the kernel parameters, edit the files `/etc/grub.d/40_custom` and `/etc/grub.d/41_custom`. After making the changes, run `update-grub` and reboot the system.

Hint: Only one `vfio-pci.ids` parameter is allowed in the kernel command line. If an existing `vfio-pci.ids` entry is already present, such as in the case of GPU passthrough, append the new PCI device ID using a comma as a separator (e.g., `vfio-pci.ids=8086:3ea5,8086:1533`).

```
sudo mousepad /etc/grub.d/40_custom

menuentry 'Hypervisor' --class gnu-linux --class gnu --class os $menuentry_
↪id_option 'rtosvisor-gnulinux-simple' {
        load_video
        gfxmode $linux_gfx_mode
        insmod gzio
        if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
        insmod part_gpt
        insmod ext2
        set root='hd0,gpt2'
        if [ x$feature_platform_search_hint = xy ]; then
          search --no-floppy --fs-uuid --set=root --hint-bios=hd0,gpt2 --
↪hint-efi=hd0,gpt2 --hint-baremetal=ahci0,gpt2  b885c987-3f50-4c4b-b242-
↪3b3c429ebde8
        else
          search --no-floppy --fs-uuid --set=root b885c987-3f50-4c4b-b242-
↪3b3c429ebde8
        fi
        linux       /boot/vmlinuz-5.15.0-88-acontis241007␣
↪root=UUID=b885c987-3f50-4c4b-b242-3b3c429ebde8 ro  quiet splash $vt_
↪handoff find_preseed=/preseed.cfg auto noprompt noefi priority=critical␣
↪locale=en_US memmap=4K\$4K amemmap_vmf=8M\$4M amemmap_shm=16M\
↪$16M memmap=256M\$256M  intel_pstate=disable acpi=force pcie_aspm.
↪policy=performance idle=poll nohalt pcie_port_pm=off pcie_pme=nomsi␣
↪cpuidle.off=1 intel_idle.max_cstate=0 nox2apic intel_iommu=on iommu=pt␣
↪intremap=off vfio_iommu_type1.allow_unsafe_interrupts=1 i915.enable_
↪psr=0 i915.enable_rc6=0 i915.enable_dc=0 i915.disable_power_well=0 i915.
↪enable_guc=2 vfio-pci.ids=8086:1533
        initrd      /boot/initrd.img-5.15.0-88-acontis241007
}
```

```
sudo update-grub
sudo reboot
```

Hint: Normally no other steps are needed to replace the native driver by the `vfio-pci` driver. If you have conflicts between these two drivers, it may be required to disable the loading of the native driver. Add

the parameter `module_blacklist=igb` to the kernel command line in that case.

Passing Device by PCI Address

If there are multiple identical PCI devices and only one should be passed to the KVM guest, we need to specify the device using its PCI address in a `udev` rule. This ensures that the selected device is detached from its default kernel driver at system startup and bound to the `vfio-pci` driver.

Create or modify a `udev` rules file for persistent binding:

```
sudo mousepad /etc/udev/rules.d/10-vfio-pci.rules
```

Insert the following rule, replacing `03:00.0` with the PCI address of your device:

```
# bind 03:00.0 to vfio-pc
ACTION=="add", SUBSYSTEM=="pci", ENV{PCI_SLOT_NAME}=="0000:03:00.0", \
RUN+="/bin/sh -c '\
    /sbin/modprobe vfio-pci && \
    echo vfio-pci > /sys/bus/pci/devices/0000:03:00.0/driver_override && \
    echo 0000:03:00.0 > /sys/bus/pci/devices/0000:03:00.0/driver/unbind &&␣
↪\
    echo 0000:03:00.0 > /sys/bus/pci/drivers/vfio-pci/bind\
'"
```

Then run these commands to make the setting effective:

```
sudo udevadm control --reload
sudo udevadm trigger --action=add --subsystem-match=pci
```

After a reboot, the device is bound to `vfio-pci`:

```
rte@rte-System-Product-Name:~$  lspci -k -s 03:00.0
03:00.0 Ethernet controller: Intel Corporation I210 Gigabit Network␣
↪Connection (rev 03)
        Kernel driver in use: vfio-pci
        Kernel modules: igb
```

VM configuration

The last step is to edit the `usr_guest_config.sh` file located in the `GUEST_FOLDER` and add the PCI Ethernet Card information here.

Uncomment `#export OTHER_HW` variable and set it to:

```
export OTHER_HW=" -device vfio-pci,host=03:00.0"
```

In case you pass through a multifunction device (e.g. 2 functions on bus 0 and device 14), then you need to add the `multifunction=on` setting:

```
export OTHER_HW=" -device vfio-pci,host=00:14.0,multifunction=on"
export OTHER_HW+=" -device vfio-pci,host=00:14.2"
```

6.11.3 Intel Integrated Graphics (iGVT-d) assignment

To use graphics passthrough, less steps compared to standard PCI hardware passthrough are required, because Hypervisor automates most of the steps.

Understanding GPU modes UPT and Legacy

There are two modes “legacy” and “Universal Passthrough” (UPT).

Hypervisor uses only Legacy mode, but it could be important to understand the difference.

UPT is available for Broadwell and newer processors. Legacy mode is available since SandyBridge. If you are unsure what processor you have, please check this link https://en.wikipedia.org/wiki/List_of_Intel_CPU_microarchitectures

In Legacy it is meant that IGD is a primary and exclusive graphics in the VM. Additionally the IGD address in the VM must be PCI `00:02.0`, only 440FX chipset model (in VM) is supported and not Q35. The IGD must be the primary GPU for Hypervisor Host as well (please check your BIOS settings).

In UPT mode the IGD can have another PCI address in VM and the VM can have a second graphics adapter (for example qxl, or vga).

Please read here more about legacy and UPT mode: https://git.qemu.org/?p=qemu.git;a=blob;f=docs/igd-assign.txt

There are a lot of other little things, why IGD Passthrough could not work. For example in legacy mode it expects an ISA/LPC Bridge at PCI address `00:1f.0` in VM and this is a reason why Q35 chip does not work, because it has another device at this address.

In UPT mode, there is no output support of any kind. So the UHD graphics can be used for accelerating (for example Decoding) but the Monitor remains black and there is a non-standard experimental qemu `vfio-pci` command line parameter `x-igd-opregion=on`, which can work.

Blocklisting the i915 driver on the Hypervisor Host

The standard `Intel Driver i915` is complex and it is not always possible to safely unbind the device from this driver, that is why this driver is blocklisted by Hypervisor when executing `/hv/bin/inithv.sh` script.

Deactivating Vesa/EFI Framebuffer on Hypervisor Host

Please also know, when i915 driver is disabled, there are other drivers which are ready to jump on the device to keep the console working. Depending on your BIOS settings (legacy or UEFI) two other drivers can occupy a region of a video memory: `efifb` or `vesafb`.

The Hypervisor blocklists both by adding the following command line parameter:

```
video=vesafb:off,efifb:off
```

Please also check if it works: `cat /proc/iomem`. If you still see that one of these drivers still occupies a part of a video memory, please try manually another combination:

```
video=efifb:off,vesafb:off.
```

Legacy BIOS, pure UEFI and CSM+UEFI in Hypervisor Host

It also plays a significant role, in which mode your machine is booted: Legacy BIOS, pure UEFI or UEFI with CSM support. In pure UEFI (on Hypervisor Host) QEMU cannot read video ROM. In this case you could extract it manually (for example using *Cpu-Z* utility or just boot in CSM mode, when iGPU is a primary GPU in BIOS), patch it with the correct device id and provide it to qemu as `romfile=` parameter for `vfio-pci`. Please google for *rom-parser* and *rom-fixer* for details.

SeaBIOS and OVMF (UEFI) in VM

It also plays a role which BIOS you use in the VM itself. For QEMU there are two possibilities: SeaBIOS (legacy BIOS, which is default for qemu) and OVMF (UEFI).

For your convenience the RTOSVisor is shipped with precompiled OVMF binaries located in the `/hv/templates/kvm` directory: `OVMF_CODE.fd` `OVMF_VARS.fd`

By default, OVMF UEFI does not support OpRegion Intel feature, which is required to have a graphics output to a real display. There are three possibilities how to solve this problem and the easiest one seems to be using special vbios rom `vbios_gvt_uefi.rom`, please read more here https://wiki.archlinux.org/index.php/Intel_GVT-g.

For your convenience, this file is already included in the Hypervisor package in the `/hv/bin` directory.

Hypervisor uses OVMF (UEFI) for graphics pass-through.

How to do it in Hypervisor

The final working configuration which we consider here:

- CPU Graphics is a primary GPU in BIOS (Hypervisor Host)
- Hypervisor Host boots in pure UEFI mode
- OVMF is used as BIOS in Windows VM
- `vbios_gvt_uefi.rom` is used as VBIOS in `romfile` parameter for `vfio-pci`
- Legacy mode for IGD, so the Windows VM has only one graphics card Intel UHD 630

Which commands should be executed in Hypervisor to do a pass-through of an Intel Integrated Graphics to a Windows VM?

None! Almost everything is done automatically. When executing `/hv/bin/inithv.sh` script (which is required to install real-time linux kernel and to reserve kernel memory for hypervisor needs), a separate GRUB entry “Hypervisor + iGVT-d” is created.

This entry already contains all necessary linux kernel parameters required to do a graphics pass-through: blocklisting intel driver, disabling interrupt remapping, assigning a VGA device to a `vfio-pci` driver and other steps.

But one step should be done once, configuring your VM.

Change `usr_guest_config.sh` script located in the `GUEST_FOLDER`. Two variables should be uncommented and activated:

```
export uefi_bios=1
export enable_vga_gpt=1
```

Just reboot your machine and choose the “Hypervisor + iGVT-d” menu item. If everything is correct, the display of your Hypervisor Host should remain black. Connect to the machine using an SSH connection or use a second graphics card for Hypervisor Host (read next chapter) and then start the guest using `hv_guest_start`

Wait 30-60 seconds and.. display remains black? Of course. Windows does not have Intel Graphics drivers.

Remember we configured Windows for a Remote Desktop Access in previous steps? Connect to Windows VM via RDP and install latest Intel Drivers https://downloadcenter.intel.com/product/80939/Graphics

If everything is done correctly, your display should now work and display a Windows 10 Desktop.

Using Second GPU Card for Hypervisor Host

X-Windows on the Hypervisor Host does not work properly, when the primary GPU in the System is occupied by the `vfio-pci` driver. It detects the first GPU, tries to acquire it, fails and then aborts. We should let it know, that it should use our second GPU card instead.

Log in to the Hypervisor Host (Press `Ctrl` + `Alt` + `F3`, for example), shutdown LightDM manager sudo service lightdm stop.

Execute

```
$ sudo X -configure
```

it creates the `xorg.conf.new` file in the current directory. When this command is executed, the X server enumerates all hardware and creates this file. By default, in modern systems, Xserver does not need the `xorg.conf` file, because all hardware is detected quite well and automatically. But the config file is still supported.

Look at this file, find a section “Device” with your second GPU (look at the PCI Adress). Copy content of this section to a separate file `/etc/X11/xorg.conf.d/secondary-gpu.conf`. Save and reboot.

```
Section "Device"
    ### Available Driver options are:-
    ### Values: <i>: integer, <f>: float, <bool>: "True"/"False",
    ### <string>: "String", <freq>: "<f> Hz/kHz/MHz",
    ### <percent>: "<f>%"
    ### [arg]: arg optional
    #Option "Accel" # [<bool>]
    #Option "SWcursor" # [<bool>]
    #Option "EnablePageFlip" # [<bool>]
    #Option "SubPixelOrder" # [<str>]
    #Option "ZaphodHeads" # <str>
    #Option "AccelMethod" # <str>
    #Option "DRI3" # [<bool>]
    #Option "DRI" # <i>
    #Option "ShadowPrimary" # [<bool>]
    #Option "TearFree" # [<bool>]
    #Option "DeleteUnusedDP12Displays" # [<bool>]
    #Option "VariableRefresh" # [<bool>]
    Identifier "Card0"
    Driver "amdgpu"
    BusID "PCI:1:0:0"
EndSection
```

Assigning an External PCI Video Card to Windows VM

External PCI Video Cards are not automatically recognized by the `/hv/bin/inithv.sh` script (unlike the CPU integrated video), so this entry is not added to the grub boot menu.

So if you want to pass the external GPU to a VM through, you first need to create a separate GRUB entry or modify an existing one.

Let's assume we've already executed the `/hv/bin/inithv.sh` script (as described in previous chapters) and it created a "Hypervisor" boot entry.

Boot computer using this "Hypervisor" entry.

Open `/boot/grub.cfg`, find its corresponding

```
menuentry 'Hypervisor'
```

section

Edit it and rename the menu entry name to:

```
menuentry 'Hypervisor + Nvidia Quadro Passthrough'
```

The kernel command line in this section should look like:

```
linux   /boot/vmlinuz-5.15.0-88-acontis root=UUID=8b4e852a-54b3-4568-a5a2-
↪dec7b16b8e07 ro  quiet splash $vt_handoff find_preseed=/preseed.cfg auto␣
↪noprompt priority=critical locale=en_US
        memmap=8k\$128k memmap=8M\$56M memmap=256M\$64M memmap=16M\$384M␣
↪maxcpus=7 intel_pstate=disable acpi=force idle=poll nohalt pcie_port_
↪pm=off pcie_pme=nomsi cpuidle.off=1
        intel_idle.max_cstate=0 noexec=off nox2apic intremap=off vfio_
↪iommu_type1.allow_unsafe_interrupts=1 intel_iommu=on iommu=pt
```

First of all we should discover the topology of PCI devices.

Find out which PCI slot is used for our external PCI card. type `lspci`.

```
00:00.0 Host bridge: Intel Corporation Device 9b53 (rev 03)
00:02.0 VGA compatible controller: Intel Corporation Device 9bc8 (rev 03)
00:08.0 System peripheral: Intel Corporation Xeon E3-1200 v5/v6 / E3-1500␣
↪v5 / 6th/7th Gen Core Proc
00:12.0 Signal processing controller: Intel Corporation Device 06f9
00:14.0 USB controller: Intel Corporation Device 06ed
00:14.1 USB controller: Intel Corporation Device 06ee
00:14.2 RAM memory: Intel Corporation Device 06ef
00:16.0 Communication controller: Intel Corporation Device 06e0
00:17.0 SATA controller: Intel Corporation Device 06d2
00:1b.0 PCI bridge: Intel Corporation Device 06c0 (rev f0)
00:1b.4 PCI bridge: Intel Corporation Device 06ac (rev f0)
00:1c.0 PCI bridge: Intel Corporation Device 06b8 (rev f0)
00:1c.4 PCI bridge: Intel Corporation Device 06bc (rev f0)
00:1c.5 PCI bridge: Intel Corporation Device 06bd (rev f0)
00:1f.0 ISA bridge: Intel Corporation Device 0685
00:1f.4 SMBus: Intel Corporation Device 06a3
00:1f.5 Serial bus controller [0c80]: Intel Corporation Device 06a4
02:00.0 VGA compatible controller: NVIDIA Corporation GM206GL [Quadro␣
↪M2000] (rev a1)
02:00.1 Audio device: NVIDIA Corporation Device 0fba (rev a1)
```

(continues on next page)

(continued from previous page)

```
04:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. Device 8125␣
↪(rev 05)
05:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network␣
↪Connection
```

Ok. `02:00.0` is our device. Let's discover, which driver occupies this device.

```
type "lspci -s 02:00.0 -vv"

02:00.0 VGA compatible controller: NVIDIA Corporation GM206GL [Quadro␣
↪M2000] (rev a1) (prog-if 00 [VGA controller])
      Subsystem: NVIDIA Corporation GM206GL [Quadro M2000]
      ...
      Kernel modules: nvidiafb, nouveau
```

So we know now which drivers should be blocklisted in kernel: `nvidiafb` and `nouveau`.

But, it looks like we have an integrated audio device in `02:00.1`. Most probably we should deactivate its driver as well.

Let's investigate our IOMMU groups, type `find /sys/kernel/iommu_groups/ -type l`:

```
/sys/kernel/iommu_groups/7/devices/0000:00:1b.0
/sys/kernel/iommu_groups/15/devices/0000:05:00.0
/sys/kernel/iommu_groups/5/devices/0000:00:16.0
/sys/kernel/iommu_groups/13/devices/0000:02:00.0
/sys/kernel/iommu_groups/13/devices/0000:02:00.1
/sys/kernel/iommu_groups/3/devices/0000:00:12.0
/sys/kernel/iommu_groups/11/devices/0000:00:1c.5
/sys/kernel/iommu_groups/1/devices/0000:00:02.0
/sys/kernel/iommu_groups/8/devices/0000:00:1b.4
/sys/kernel/iommu_groups/6/devices/0000:00:17.0
/sys/kernel/iommu_groups/14/devices/0000:04:00.0
/sys/kernel/iommu_groups/4/devices/0000:00:14.1
/sys/kernel/iommu_groups/4/devices/0000:00:14.2
/sys/kernel/iommu_groups/4/devices/0000:00:14.0
/sys/kernel/iommu_groups/12/devices/0000:00:1f.0
/sys/kernel/iommu_groups/12/devices/0000:00:1f.5
/sys/kernel/iommu_groups/12/devices/0000:00:1f.4
/sys/kernel/iommu_groups/2/devices/0000:00:08.0
/sys/kernel/iommu_groups/10/devices/0000:00:1c.4
/sys/kernel/iommu_groups/0/devices/0000:00:00.0
/sys/kernel/iommu_groups/9/devices/0000:00:1c.0
```

so, we see, that our NVidia card belongs to a IOMMU group 13, together with its audio device.

So, repeat steps for audio device, type `lspci -s 02:00.1 -vv`:

```
02:00.1 Audio device: NVIDIA Corporation Device 0fba (rev a1)
...
Kernel modules: snd_hda_intel
```

Now all these `3` drivers should be blocklisted in your system.

Add the following to your "Hypervisor" entry kernel command line: `module_blacklist=nouveau,nvidiafb,snd_hda_intel`

Secondly, assign devices to the KVM.

In order to make it possible to pass NVidia VGA and Audio device to a Windows VM through we should assign a special driver `vfio-pci` to each of our devices.

Let's determine Vendor and Device IDs:

```
type "lspci -s 02:00.0 -n"
02:00.0 0300: 10de:1430 (rev a1)

type "lspci -s 02:00.1 -n"
02:00.1 0403: 10de:0fba (rev a1)
```

Add the following to our kernel command line: `vfio-pci.ids=10de:1430,10de:0fba`

So our final command line should now look like:

```
linux   /boot/vmlinuz-5.15.0-88-acontis root=UUID=8b4e852a-54b3-4568-a5a2-
↪dec7b16b8e07 ro  quiet splash $vt_handoff find_preseed=/preseed.cfg auto␣
↪noprompt priority=critical locale=en_US
        memmap=8k\$128k memmap=8M\$56M memmap=256M\$64M memmap=16M\$384M␣
↪maxcpus=7 intel_pstate=disable acpi=force idle=poll nohalt pcie_port_
↪pm=off pcie_pme=nomsi cpuidle.off=1
        intel_idle.max_cstate=0 noexec=off nox2apic intremap=off vfio_
↪iommu_type1.allow_unsafe_interrupts=1 intel_iommu=on iommu=pt module_
↪blacklist=nouveau,nvidiafb,snd_hda_intel
        vfio-pci.ids=10de:1430,10de:0fba
```

If you boot the computer into this new grub entry, you could check if the `vfio-pci` driver successfully acquired our devices:

```
type "dmesg | grep vfio"

[    5.587794] vfio-pci 0000:02:00.0: vgaarb: changed VGA decodes:␣
↪olddecodes=io+mem,decodes=io+mem:owns=none
[    5.606445] vfio_pci: add [10de:1430[ffffffff:ffffffff]] class 0x000000/
↪00000000
[    5.626451] vfio_pci: add [10de:0fba[ffffffff:ffffffff]] class 0x000000/
↪00000000
```

Everything is ok.

Thirdly, check if your PCI Card is not a primary graphics device in your system. Boot into BIOS and find the corresponding settings. For every BIOS there are own namings for this option.

Often there is no such option in the BIOS and it then detects which HDMI ports are connected and if both Integrated GPU and the external PCI card are connected to monitors, then the integrated card is chosen by BIOS as a primary device. If not, then your PCI card is selected as the primary device.

If your card is not a primary device and integrated CPU graphics is used for Hypervisor Host, skip reading this section. But if not, you should disable frame buffer, because its drivers occupy video card PCI regions.

Add `video=efifb:off,vesafb:off disable_vga=1` to your kernel command line.

This trick is also useful, when your PC has no integrated CPU graphics and the external PCI Video card is the only device in your system.

As the fourth step, assign your VGA and Audio devices to your Windows VM (Qemu)

Usually the VM configuation is located in this file: `GUEST_FOLDER/usr_guest_config.sh`

Find and change `OTHER_HW` line from

```
export OTHER_HW=$OTHER_HW
```

to

```
export OTHER_HW=$OTHER_HW" -device vfio-pci,host=02:00.0 -device vfio-pci,
↪host=02:00.1 -nographic"
```

Note: Please note, you need to install your video card manufacturer drivers to a Windows VM, to make the graphics working.

Create a temporary VGA device or a Windows RDP/Qemu VNC connection to your VM to install graphics drivers.

6.11.4 Keyboard and Mouse assignment

Normally your Hypervisor Host with Windows VM and Integrated Graphics passed through works in headless mode. Windows VM outputs to a monitor via a DVI-D/HDMI connection and the Hypervisor Host is controlled via an SSH connection. Windows has a look and feel as it works without an intermediate hypervisor layer.

So, Windows needs a keyboard and mouse.

Go to `/dev/input/by-id/` and find something that looks like a keyboard and mouse and it should contain **"-event-"** in its name.

```
ls -la /dev/input/by-id | grep -event-

usb-18f8_USB_OPTICAL_MOUSE-event-if01 -> ../event5
usb-18f8_USB_OPTICAL_MOUSE-event-mouse -> ../event3
usb-18f8_USB_OPTICAL_MOUSE-if01-event-kbd -> ../event4
usb-SEM_USB_Keyboard-event-if01 -> ../event7
usb-SEM_USB_Keyboard-event-kbd -> ../event6
```

Configure your VM with these parameters by editing `usr_guest_config.sh`:

```
export vga_gpt_kbd_event_device="usb-SEM_USB_Keyboard-event-kbd"
export vga_gpt_mouse_event_device="usb-18f8_USB_OPTICAL_MOUSE-event-mouse"
```

Please also note, disconnecting evdev devices, such as keyboard or mouse, can be problematic when using qemu and libvirt, because it does not reopen the device when the device reconnects.

If you need to disconnect/reconnect your keyboard or mouse, there is a workaround: create a udev proxy device and use its event device instead. Please read more here: https://github.com/aiberia/persistent-evdev.

If everything works, you'll find new devices like `uinput-persist-keyboard0` pointing to `/dev/input/eventXXX`. Use these device names in:

```
export vga_gpt_kbd_event_device="uinput-persist-keyboard0"
export vga_gpt_mouse_event_device="mouse-device-name"
```

6.12 Start a Real-time Application

6.12.1 Start RT-App in Windows guest

In this section we will show how to execute an application on the RT-Linux guest from within the Windows guest.

Caution: It is assumed that

- all the provided desktop shortcuts have been placed on the desktop, see last chapter.
- RT-Linux was started before the Windows guest.

- Start the Windows guest
- Download and install the appropriate putty package from https://www.putty.org/
- Execute the *"Hypervisor Attach"* shortcut
- Open putty and connect to `192.168.157.2`, accept the connection (do **not** login)
- Close putty (we only need to execute this step to accept the security keys, they are needed to run the RT-Linux application)
- Execute the *"Run RT-Linux Application"* shortcut
- A message should be shown.

Hint: This shortcuts executes the `"remote_cmd_exec"` sample script located in `/hv/guests/guestrtlinux/files/remote_cmd_exec.sh`.

You may adjust this file according to your needs.

6.12.2 Start RT-App in Linux guest

In this section we will show how to execute an application on the RT-Linux guest from within the Windows guest.

Caution: It is assumed that

- all the steps described in `Hypervisor Ubuntu Guest` have been executed.
- RT-Linux was started before the Ubuntu guest.

- Start the Ubuntu guest
- Execute the *"Hypervisor Attach"* shortcut or call `hv_attach` in console
- Execute `ssh root@192.168.157.2` in console
- Log into `Real-Time Linux`:

```
$ vmf64 login: root
$ password:  root
```

7 KVM Guest Network

There are various options how to configure a KVM guest for networking.

- Private network between the guest and the Hypervisor Host (e.g. for file sharing)
- Virtual network provided by the communication subsystem to communicate between KVM guests and RTOS guests via TCP/IP.
- One single dynamically bridged automatic network connection to the external network (the network adapter typically is shared between all guests and the hypervisor).
- Network adapter statically bridged with a physical adapter in the Hypervisor Host (typically only used by a single guest).
- Network adapter statically bridged with the virtual network (if the communication subsystem shall not be used by the guest).
- Network adapter dynamically bridged with the virtual network (if the communication subsystem shall not be used by the guest).
- Guest internal statically bridged network (KVM guests can communicate over this internal network without using a physical ethernet controller and without using the virtual network).

The network configuration is based on various settings provided in `usr_guest_config.sh` in the `GUEST_FOLDER`. See details below.

If one or multiple network configurations are enabled using a static bridge, such a bridge is created once when the guest is started. This bridge will then be preserved until the system is rebooted.

To verify which bridges are available, run:

```
$ brctl show
```

You can remove these bridges (after stopping all guests) using the command `hv_brdelall`:

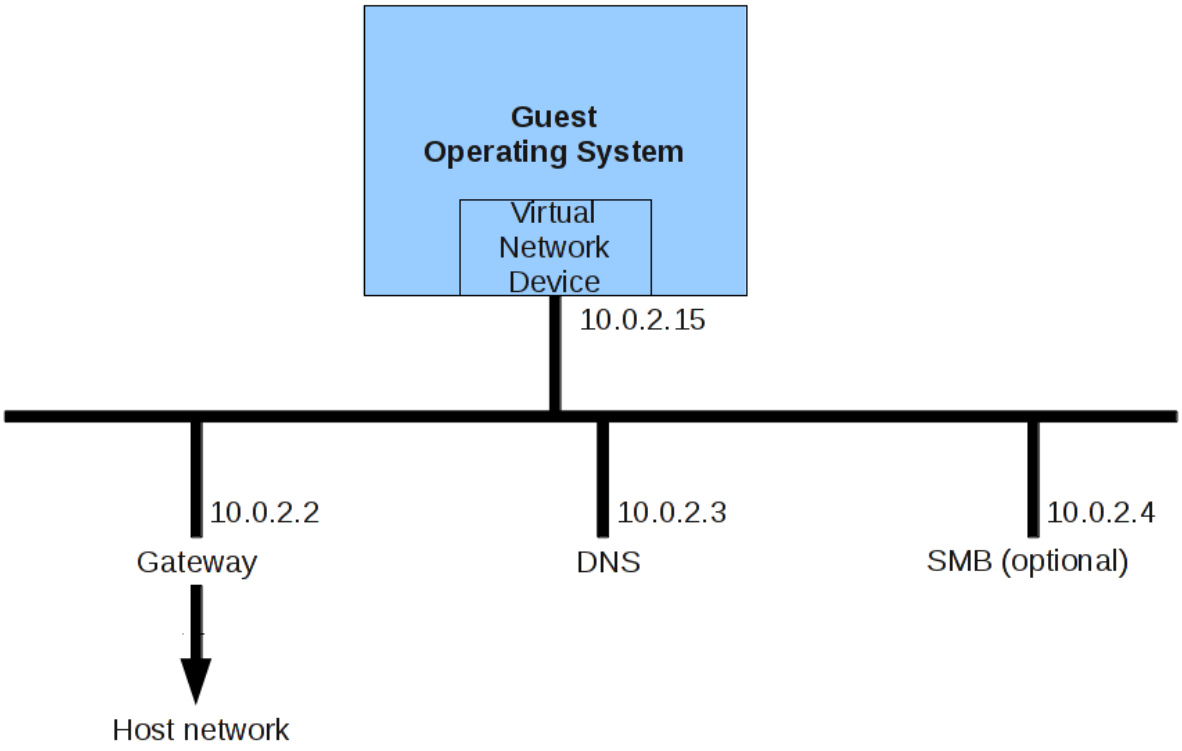
```
$ hv_brdelall
```

Caution: All static bridges are created once when a guest is started. If you stop the guest and change related configuration you need to remove the related bridges, otherwise the changes will not become effective and the behaviour may become undefined. To accomplish this, just stop all guests and remove all bridges:

```
$ hv_brdelall
```

7.1 Private guest/Hypervisor network

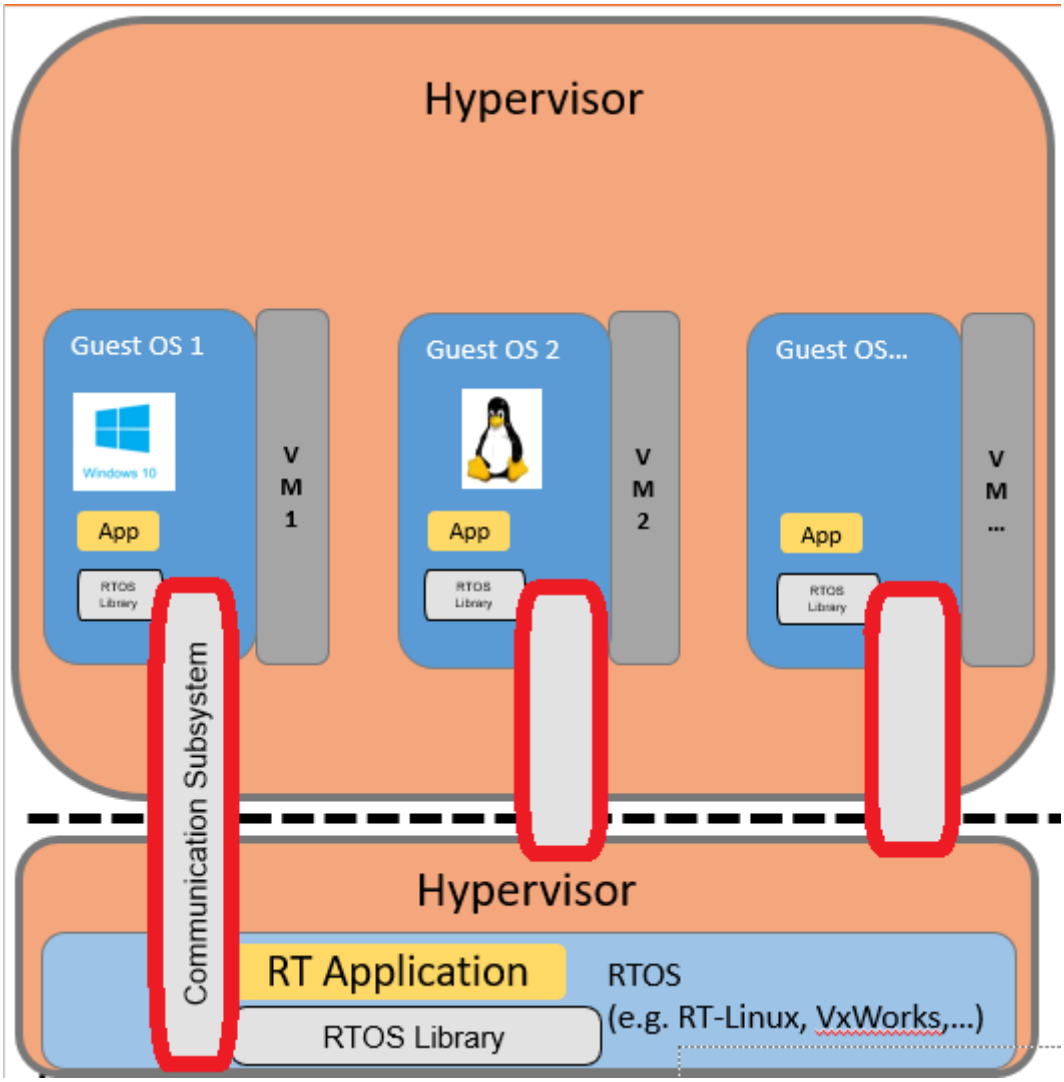
The private network is a Hypervisor internally provided network which is connected to the external physical network via NAT (if the Hypervisor Host is connected to the external network). This connection is safe because it cannot be accessed from the outside, but it is also much slower. It is also used for file sharing with the Hypervisor Host. The architecture of the private network looks as follows:



By default, the SMB server is enabled. This server provides access to the Hypervisor Host file system. For details, see *Default SMB share*

7.2 Communication subsystem Virtual Network

The communication subsystem is provided by the hypervisor to support guest communication. One of the communication means is the Virtual Network.



To use the communication subsystem in KVM guests, the hypervisor support package needs to be installed. Before using the communication subsystem, the KVM guest has to `attach` to the Virtual Machine Framework (VMF) using the Uploader tool. After installation of the support package, a respective shortcut should be available on the desktop.

The IP addresses of the virtual network are set to fixed values.
The default IP addresses are:
Hypervisor Host: 192.168.157.1
First RTOS guest: 192.168.157.2
Windows example guest: 192.168.157.3

You have to set the IP addresses as well as the MAC addresses in the guests to unique values.

7.3 Automatic external network connection (dynamically bridged)

The network settings are provided in `usr_guest_config.sh` in the `GUEST_FOLDER`. One single automatic network connection to the external network is provided. To enable this network, set `external_nw` to a value of `1`. This network typically is shared between all guests and the hypervisor.

The `netif_mode` setting determines, how this network is set up.

- 0: manual parameter setting. The parameters `netif_m`, `defaultgw_m`, `dns_m`, `brip_m` and `brnm_m` have to be set properly. The Hypervisor Host network settings also need to be set manually then.
- 1: automatic parameter setting. This is the default setting and should be used if the network where the hypervisor is connected to supports DHCP.
- 2: no IP for Hypervisor Host. The parameter `netif_m` has to be set properly. In this case, the Hypervisor Host has no IP connection via the adapter defined in `netif_m`.

Caution: If you have configured the network for automatic mode (1), the DHCP server must be configured to always provide the same IP address to the hypervisor. While the KVM guest is running, the lease will never be re-requested and after the KVM guest has shut down, the IP address is re-assigned to the hypervisor physical network.

You may have to set the Hypervisor Host network adapter accordingly. See *Hypervisor Host network configuration* for details.

When the guest is started, a bridge is dynamically created in the Hypervisor Host. This will bridge the physical device with a so called `tap` device representing the guest. After stopping the guest, this bridge is removed.

The name of the statically created bridge will be `vmbridge`.

7.4 Bridged external network connection (static bridge)

The network settings are provided in `usr_guest_config.sh` in the `GUEST_FOLDER`. One or multiple physical network adapters can be used in the guest using a (static) bridge. Using this method, multiple network adapters can be assigned to guests. Typically in such a use case, a single network adapter is only used by one single guest (not used by the Hypervisor Host nor shared by multiple guests).

To enable this mode, you need to set `phys_nw` to a value of `1` and enable one or multiple entries `physnw_dev` and `physnw_ip`. For each single network adapter all two values need to be provided.

The name of the statically created bridge will be `br-DEVICE`, for example `br-enp1s0` for the `enp1s0` device.

7.5 Bridged virtual network connection (static bridge)

The network settings are provided in `usr_guest_config.sh` in the `GUEST_FOLDER`. If you want to communicate between a KVM guest and an RTOS guest, you can use the virtual network. This is either possible using the communication subsystem as described above. Alternatively, a static bridge can be used, then avoiding the need to use the communication subsystem.

You need to set `rtosvnet_nw` to a value of `1` to accomplish this.

The name of the statically created bridge will be `br-rtosvnet`.

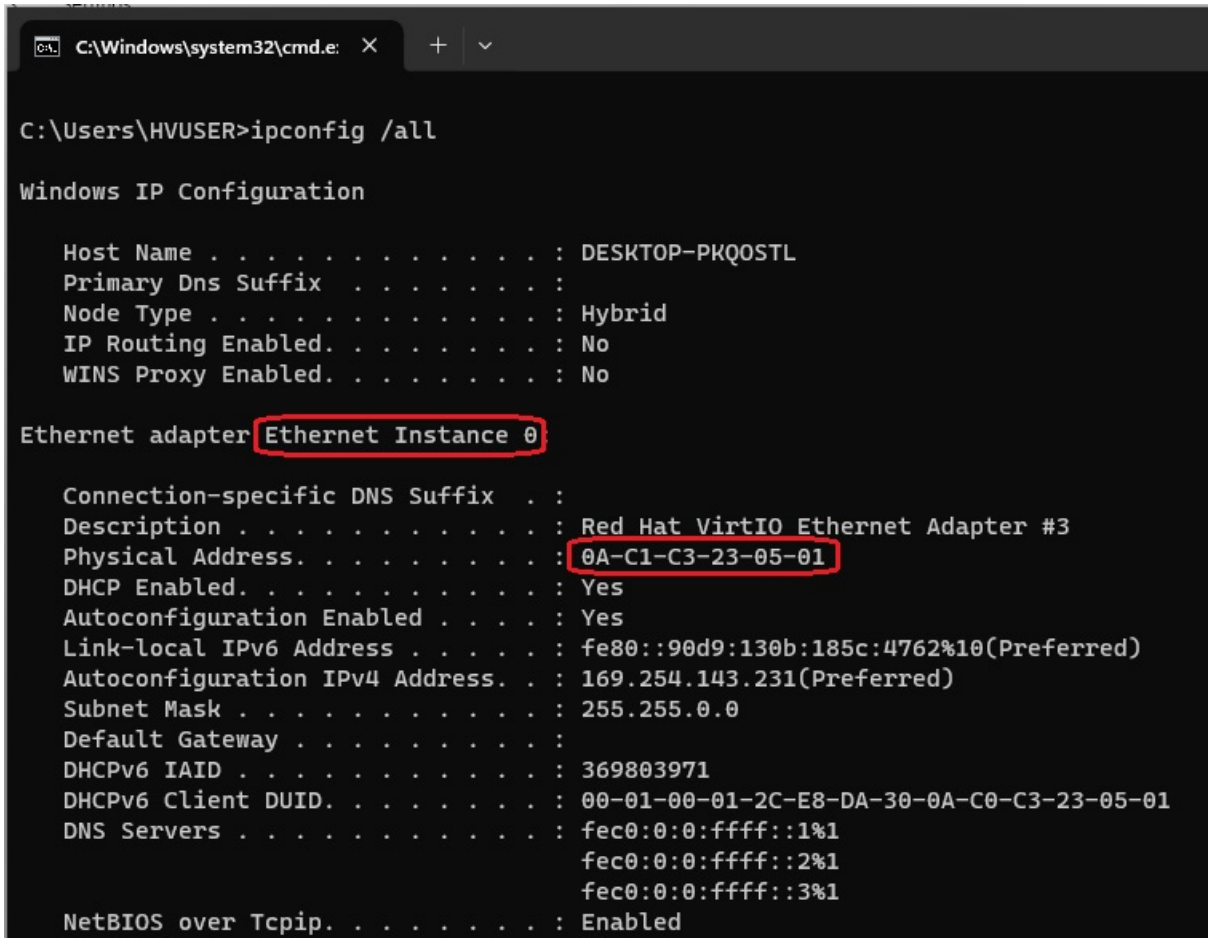
To establish a connection from the KVM guest with the virtual network, an IP address must be assigned to the corresponding Ethernet adapter in the KVM Guest. To modify the address of the virtual network, you can change the value of `rtosvnet_br_ip` in the configuration file `usr_guest_config.sh`. By default, the address is set to `192.168.157.1`.

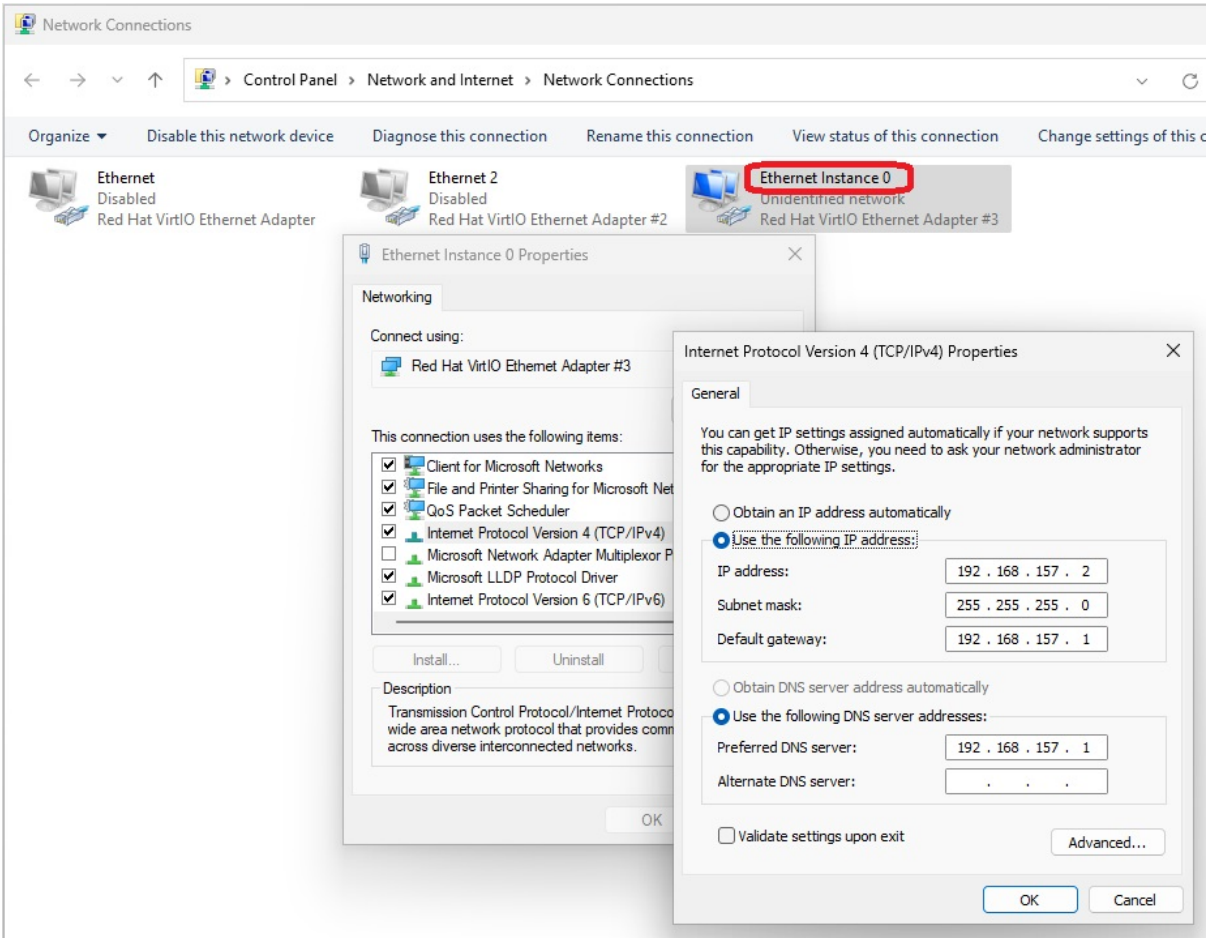
Here is an example for a Windows guest:

Query the MAC address with `brctl showmacs` **after** the Windows guest has finished booting.

```
$ $ brctl showmacs br-rtosvnet
$ port no  mac addr            is local?    ageing timer
$   1      00:60:c8:00:00:00   yes          0.00
$   1      00:60:c8:00:00:00     yes          0.00
$   2      0a:c1:c3:23:05:01     no           0.13          #---- MAC address
 ↪  needed for configuration
$   2      ea:e8:fd:54:69:ee     yes          0.00
$   2      ea:e8:fd:54:69:ee     yes          0.00
```

Use the MAC address with the ‘is local = no’ entry for configuring the network card. The network card in the KVM guest can be identified by running `ipconfig /all` in the console. Set the network address in the Windows `Network Connections`.


C:\Windows\system32\cmd.e:
C:\Users\HVUSER>ipconfig /all
Windows IP Configuration
Host Name . . . . . : DESKTOP-PKQOSTL
Primary Dns Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
Ethernet adapter Ethernet Instance 0
Connection-specific DNS Suffix . :
Description . . . . . : Red Hat VirtIO Ethernet Adapter #3
Physical Address. . . . . : 0A-C1-C3-23-05-01
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
Link-local IPv6 Address . . . . . : fe80::90d9:130b:185c:4762%10(Preferred)
Autoconfiguration IPv4 Address. . : 169.254.143.231(Preferred)
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . :
DHCPv6 IAID . . . . . : 369803971
DHCPv6 Client DUID. . . . . : 00-01-00-01-2C-E8-DA-30-0A-C0-C3-23-05-01
DNS Servers . . . . . : fec0:0:0:ffff::1%1
fec0:0:0:ffff::2%1
fec0:0:0:ffff::3%1
NetBIOS over Tcpip. . . . . : Enabled



Caution: By default, the Windows firewall prevents responding to a ping from the Hypervisor Host. You must disable the firewall or set up an appropriate exception to be able to ping from the Hypervisor Host to the KVM guest.

7.6 Bridged virtual network connection (dynamic bridge)

The network settings are provided in `usr_guest_config.sh` in the `GUEST_FOLDER`. If you want to communicate between a KVM guest and an RTOS guest, you can use the virtual network. This is either possible using the communication subsystem as described above. Alternatively, a dynamic bridge can be used, then avoiding the need to use the communication subsystem.

You need to set `vnet_nw` to a value of `1` to accomplish this.

The name of the dynamically created bridge will be `vmvnetbridge`.

Caution: If you have configured static bridges you have to remove all bridges before setting up the dynamic bridge:

```
$ hv_brdelall
```

To establish a connection from the KVM guest to the virtual network, an IP address must be assigned to the corresponding Ethernet adapter in the KVM guest.

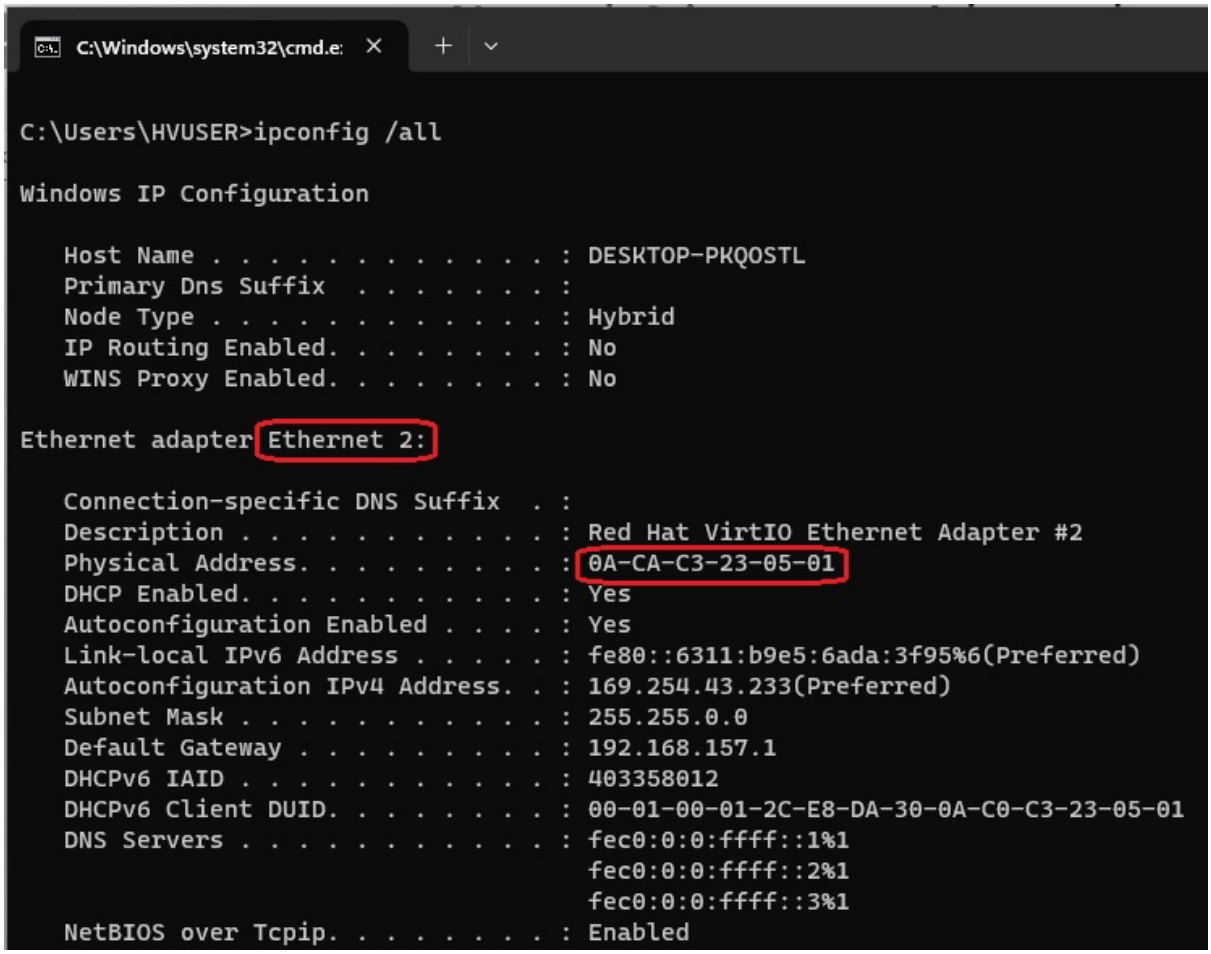
Here is an example for a Windows guest:

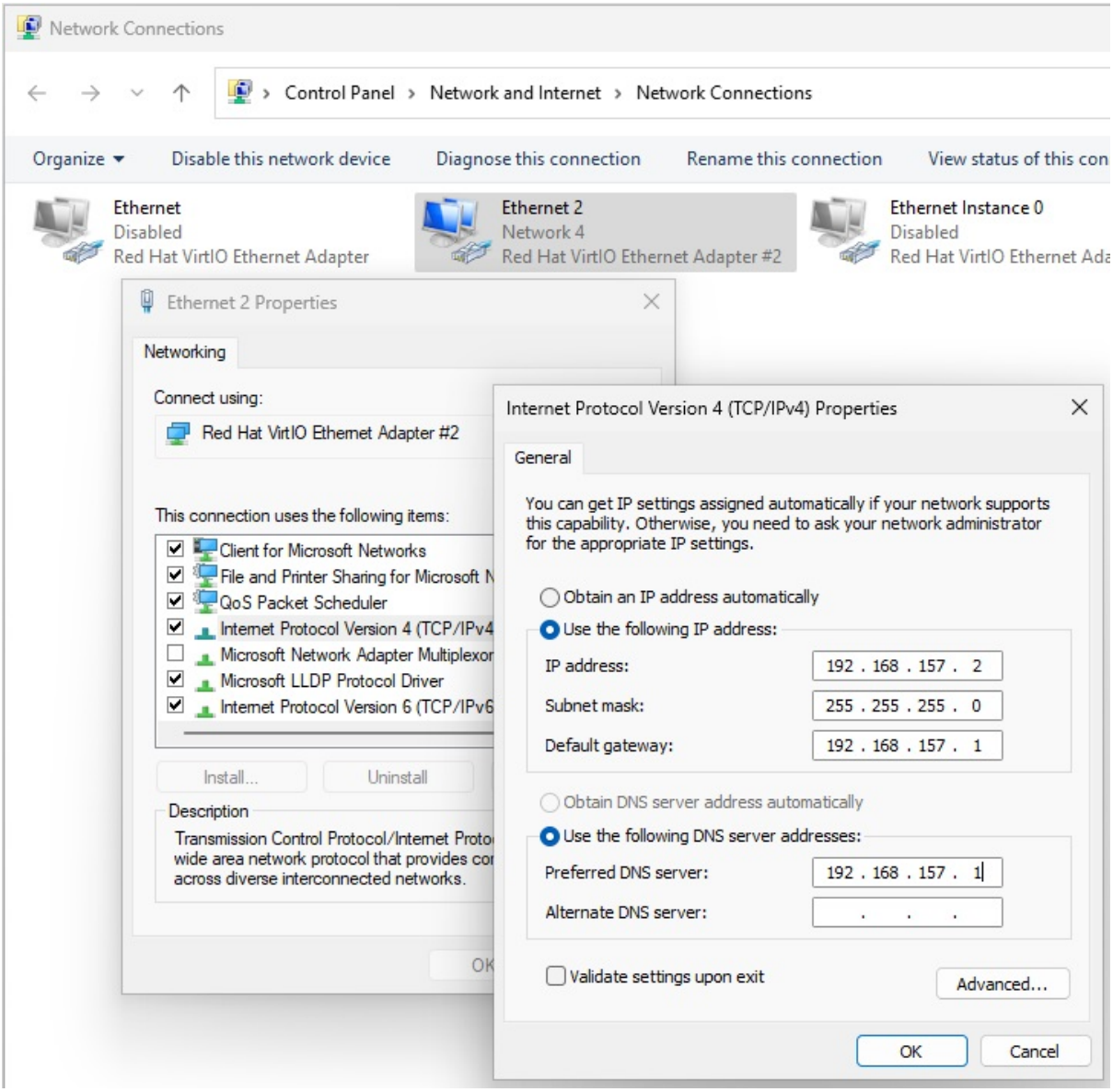
You will find the MAC address in the output when starting the KVM guest.

```
$ $ hv_guest_start -view
$ verify and prepare VM parameters
$ external network MAC = 0A:C0:C3:23:05:01
$ vnet network MAC = 0A:CA:C3:23:05:01     #---- virtual Network MAC needed
 ↪  for configuration
$ private network MAC = 0A:C1:C3:23:05:01
$ start VM
```

Use the `vnet network MAC` address for configuring the network card in the KVM guest. The corresponding network card can be identified by running `ipconfig /all` in the console. The network address is then set in the Windows `Network Connections`. By default, the address of the Hypervisor Host in the virtual network is 192.168.157.1. You can verify the IP Adress on the Hypervisor Host:

```
$ $ ip addr show vmvnetbridge
$    vmvnetbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
 ↪  state UP group default qlen 1000
$     link/ether 00:60:c8:00:00:00 brd ff:ff:ff:ff:ff:ff
$     inet 192.168.157.1/24 brd 192.168.157.255 scope global vmvnetbridge
```





Caution: By default, the Windows firewall prevents responding to a ping from the Hypervisor Host. You must disable the firewall or set up an appropriate exception to be able to ping from the Hypervisor Host to the KVM guest.

7.7 KVM guest internal network (static bridge)

If you want to communicate between a KVM guest and an RTOS guest, you can use the virtual network. For guest internal TCP/IP communication it is possible to set up a guest internal network. This network has no connection to the outside physical network.

You need to set `guest_nw` to a value of `1` to enable this network. The Hypervisor Host may also be connected to this network by setting the `guest_br_ip` parameter to an appropriate IP address and network mask. In the default case, the Hypervisor Host is not connected to this network (`guest_br_ip` set to `no`).

The name of the statically created bridge will be `br-guest`.

7.8 Guest Ethernet MAC addresses

In the `GUEST_FOLDER` the script `GUEST_NAME_setmac.sh` defines the Ethernet MAC address for the virtual network adapters inside the guest. By default, if this file does not exist, this script file is automatically generated using random local administered addresses. You will have to change these addresses by an official address related to your company.

The content of this file looks as follows:

```
$ export ethmacVM1=0A:C0:3E:89:08:04
$ export ethmacVM2=0A:C1:3E:89:08:04
$ export ethmacVM3=0A:C2:3E:89:08:04
$ export ethmacVM4=0A:C3:3E:89:08:04
$ export vnetethmacVM1=0A:CA:3E:89:08:04
$ export ethmacPHYS
$ ethmacPHYS[0]=0A:D0:3E:89:08:04
$ ethmacPHYS[1]=0A:D1:3E:89:08:04
$ ethmacPHYS[2]=0A:D2:3E:89:08:04
$ ethmacPHYS[3]=0A:D3:3E:89:08:04
$ ethmacPHYS[4]=0A:D4:3E:89:08:04
$ ethmacPHYS[5]=0A:D5:3E:89:08:04
$ ethmacPHYS[6]=0A:D6:3E:89:08:04
$ ethmacPHYS[7]=0A:D7:3E:89:08:04
$ ethmacPHYS[8]=0A:D8:3E:89:08:04
$ ethmacPHYS[9]=0A:D9:3E:89:08:04
```

The `ethmacVM1` address belongs to the bridged Ethernet controller (bridged to the external network).

The `ethmacVM2` address belongs to the private (internal) Ethernet controller (using NAT).

The `ethmacVM3` address belongs to the statically bridged Virtual network.

The `ethmacVM4` address belongs to the guest internal network.

The `vnetethmacVM1` address belongs to the dynamically bridged Virtual network.

The `ethmacPHYS[#]` addresses belong to the statically bridged external network (if MAC replication is turned off).

8 SDKs

Each part of the hypervisor supplies one or more SDKs, which can be used by custom-compiled applications. The SDKs can be categorized by Hypervisor Host, realtime guest and general purpose guest SDKs.

8.1 Hypervisor Host SDK

Applications which are running at the **Hypervisor Host** and should communicate with the guests, need the `Hypervisor Host SDK` as api.

The `Hypervisor Host SDK` is located in the `/hv/sdk` folder. It contains currently an `/inc`, a `/lib` and an `/examples` sub-directory.

8.2 Windows Guest SDK

The SDKs are available in the **Windows** guest **after** installing `RTOSVisor.exe`.

- Install `/hv/guests/files/RTOSVisor.exe`
- Location of the SDK directory: `%Program Files%\acontis_technologies\Hypervisor\SDK`

Hint: The supplied product `zip` package contains also this SDK at `/Hypervisor/SDK/...`.

8.3 Linux Guest SDK

The SDKs are available **after** installing the `hvUbuntuGuestSdkPackage_x.x.xx.xx_amd64.deb` package. This is done by calling the installation script `install_sdk.sh`. Both files are located on the Hypervisor Host in the directory `/hv/guests/files/LinuxTools`.

It is possible to install the SDK within the Linux guest or a separate development PC with Linux installed.

8.3.1 Install SDK in Linux guest

The installation is provided in a mountable directory on the Hypervisor Host. To access this directory from the Linux guest we need to install the network client software `cifs-utils` and mount the SMB drive from QEMU where the installation is located.

Open a `console` prompt in the Ubuntu guest and enter the following:

```
$ sudo apt-get update
$ sudo apt-get install cifs-utils
$ sudo mkdir /mnt/qemu
$ sudo mount -t cifs //10.0.2.4/qemu /mnt/qemu -o guest
$ sudo /mnt/qemu/files/LinuxTools/install_sdk.sh
```

With the installation process, the directory `/hv/sdk` is created. This is where all the essential files required for the SDK are stored.

8.3.2 Install SDK on a Linux development PC

To install the SDK on a separate development PC the `hvUbuntuGuestSdkPackage_x.x.xx.xx_amd64.deb` and `install_sdk.sh` files have to be copied from the Hypervisor Host `/hv/guests/files/LinuxTools` to the development PC. When called, the script `install_sdk.sh` will install the SDK in the directory `/hv/sdk`.

8.4 VxWorks

Please contact support.

8.5 RTOS-32

There are separate tutorials which describe how to set up remote debugging for On Time RTOS-32.

9 Automatic Startup and Shutdown

9.1 Hypervisor Host autologin

If you want to enable automatic login to the Hypervisor Host after booting has finished, follow the next steps.

```
$ sudo -e mousepad
 ↪  /usr/share/lightdm/lightdm.conf.d/60-lightdm-gtk-greeter.conf
```

```
# autologin-settings to be appended in Section [Seat:*]
autologin-user=insert_your_username_here
autologin-user-timeout=0
```

9.2 Automatic guest startup

9.2.1 Enable autostart

In this section we will show how one or multiple guests can be started automatically after the Hypervisor has finished booting.

Introduction

To automatically start guests after the hypervisor finished booting, you need to adjust the `/hv/config/usr_guest_autostart.config` file.

```
$ mousepad /hv/config/usr_guest_autostart.config
```

All guests that shall be started automatically have to be included here by inserting their respective guest foldername. The following example configuration will automatically start `4` guests, located in the folders `guest0001` to `guest0004`.

```
/hv/guests/guest0001
/hv/guests/guest0002
/hv/guests/guest0003
/hv/guests/guest0004
```

The configuration settings will only become effective after running the command `hv_set_autostart`. This command requires the hypervisor root configuration file name as parameter. The root configuration file used by the System Manager and the examples is located in `/hv/config/hv.config`.

```
$ hv_set_autostart -enable /hv/config/hv.config
```

Hint:

This command will copy, rename and adjust the `hv_guest_autostart.service` template located in the guest folders into the `/hv/services/hv_guest_autostart-GUEST.service` (where GUEST is the guest foldername).
Then it will enable a respective `systemd` service located in `/etc/systemd/system`.

This service will automatically start the guest after booting the system.

Caution: If you change the settings in `usr_guest_autostart.config` you need to first disable the autostart configuration and then re-enable the new configuration:

```
$ hv_set_autostart -disable
$ hv_set_autostart -enable /hv/config/hv.config
```

Automatic startup of example guests (RT-Linux)

Hint: Initially, the Hypervisor Host does not provide any example guest folders. To switch to the RT-Linux guest example, you need to perform the suitable initialization. For instructions on initializing the examples, please see the **RTOS Guests** chapter in the Hypervisor Manual.

```
hv_open_example rt-linux
hv_sync_example rt-linux
cd /hv/guests/guestrtlinux
```

Uncomment the line `/hv/guests/guestrtlinux show_console` in `usr_guest_autostart.config` to automatically configure starting the example RT-Linux guest. You may also uncomment the line `/hv/guests/guestwindows show_console` to automatically configure starting the example virtual KVM guest (e.g. Windows or Ubuntu). In this case, you should initialize the associated example.

```
;#####################################################
; user specific guest autostart configuration script
;#####################################################

; autostart example RT-Linux guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/guestrtlinux show_console

; autostart example Windows guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/guestwindows show_console

; autostart example Ubuntu guest
; if guest console shall be shown automatically, use parameter show_console
; /hv/guests/guestubuntu show_console

; autostart system manager created guests (to be removed if system manager␣
↪supports autostart handling)
; if guest console shall be shown automatically, use parameter show_console
```

Then enable autostart:

```
$ hv_set_autostart -enable /hv/config/hv.config
```

Automatic startup of System Manager managed guests

Currently, the System Manager does not support setting autostart configuration from within the GUI. It is required to add additional lines `/hv/guests/guestXXXX show_console` in `usr_guest_autostart.config` to automatically configure starting System Manager generated guests.

```
;#################################################
; user specific guest autostart configuration script
;#################################################

; autostart example RT-Linux guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/guestrtlinux show_console

; autostart example Windows guest
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/guestwindows show_console

; autostart example Ubuntu guest
; if guest console shall be shown automatically, use parameter show_console
; /hv/guests/guestubuntu show_console

; autostart system manager created guests (to be removed if system manager␣
↪supports autostart handling)
; if guest console shall be shown automatically, use parameter show_console
/hv/guests/guest0001
/hv/guests/guest0002
```

When enabling autostart, the `/hv/config/hv.config` file needs to be used:

```
$ hv_set_autostart -enable /hv/config/hv.config
```

Standard console mode

The `show_console` parameter following the guest foldername will automatically launch a console window to the guest when logging in into the Hypervisor Host.

The following example configuration will automatically start `3` guests, located in the folder `guest0001` to `guest0003`. A console window will be displayed for guest `1` and `3`.

```
/hv/guests/guest0001 show_console
/hv/guests/guest0002
/hv/guests/guest0003 show_console
```

Hint:

Here, the `hv_guest_autostart_xfce.sh` template will also be copied, renamed and adjusted into `/hv/services/hv_guest_autostart_xfce-GUEST.sh`.
When rebooting and after logging in, the Hypervisor will read the `autostart.desktop` file located in the home folder in `~/.config/autostart`. This file points to a script file `/hv/bin/xfce-hv_autostart.sh` which launches all console windows that are configured in `usr_guest_autostart.config`.

Standalone console mode

The `standalone_console` parameter following the guest foldername will automatically launch a full screen console window to the guest when logging in into the Hypervisor Host.

The following example configuration will automatically start `3` guests, located in the folder `guest0001` to `guest0003`. A full screen console window will be displayed for guest `3`.

```
/hv/guests/guest0001
/hv/guests/guest0002
/hv/guests/guest0003 standalone_console
```

By default, the monitor/display is configured to never turn off.
This can be adjusted by editing the script
`/hv/bin/guest_autostart_standalone_dispconf.sh` and commenting out the respective commands.

```
# disable DPMS (Energy Star) features
# sudo xset -dpms >>$HV_BIN/guest_autostart_standalone_dispconf.log 2>&1

# turn off screen saver
# sudo xset s off >>$HV_BIN/guest_autostart_standalone_dispconf.log 2>&1
```

Caution: You should disable the Hypervisor Host desktop to assure the standalone console mode works properly. See section *Disable Hypervisor Host desktop* how to disable the Hypervisor Host desktop.

Caution: In the current RTOSVisor version, the `standalone_console` only works for the guest which is started last.

Caution: The `standalone_console` must not be used in conjunction with graphics passthrough.

Hint:

This command will copy, rename and adjust the `hv_guest_autostart_standalone.service` template located in the guest folders into the `/hv/services/hv_guest_autostart_standalone-GUEST.service` (where GUEST is the guest foldername).
Then it will enable a respective `systemd` service located in `/etc/systemd/system`.
This service will automatically start the guest after booting the system.

Caution: Only one single KVM guest (no RTOS guest) can use the standalone console. Furthermore, if one guest is using the standalone console, no other guests can use the standard console anymore.

9.2.2 Disable autostart

To disable automatic start of guests, run the command

```
$ hv_set_autostart -disable
```

A complete reset of the autostart configuration can be done using the following command (it will not change the configuration settings in `usr_guest_autostart.config`):

```
$ hv_set_autostart -reset
```

9.2.3 Verify autostart

To show the current autostart configuration, run the command

```
$ hv_set_autostart -show
```

9.3 Hypervisor Host services

The hypervisor is using systemd services to automatically start guests, drivers etc. All services are located in the `/hv/services` folder. Active services will have a link set in `/etc/systemd/system` which point to the respective service file in `/hv/services`

The following services exist:

```
hv_initial_cleanup.service      - run initial cleanup tasks (e.g. in case
 ↪  of a non graceful shutdown)
hv_netif_restore.service        - restore network settings after reboot (in
 ↪  case KVM guests were not graceful shutdown)
hv_system_observer.service      - handles graceful system shutdown or reboot
hv_diskshare_observer.service   - support for dynamic disk forwarding to
 ↪  KVM guests (e.g. USB sticks)
hv_sysmgr.service               - start the graphical System Manager web
 ↪  server backend
hv_loaddriver.service           - load the basic hypervisor drivers
hv_part.service                 - hypervisor partitioning, e.g. assigns PCI
 ↪  Ethernet Cards to a RTOS
hv_ifup_vnet0.service           - bring up the virtual network device
hv_usbip_expose.service         - expose USB devices to Real-time guests
hv_vmf_autostart.service        - start the Virtual Machine Framework (the
 ↪  RTOS-VM Hypervisor) and load the hypervisor configuration
hv_guest_autostartXxxx.service  - start one specific guest
```

9.4 Autostart RT-Linux Applications

Automatic startup of RT-Linux applications is configured in the `guest.config` file located in the guest folder (section `[Rtos\Autostart\1]`). By default automatic start is enabled.

If this autostart section is active, the autostart script `autostart.sh` will be executed after RT-Linux has booted. This file is located in the `files` subfolder of the guest folder.

You may adjust both files according to your needs.

9.5 System Shutdown and Power-off/Reboot

When the system shall be powered off or rebooted, it is required to savely shutdown all guests before. Every time when a power off or reboot sequence is initiated, the Hypervisor host will assure that before it is shutting down, all the guests will be terminated first. This is handled by the `hv_system_observer.service` service which calls the `hv_observer` command to manage the system termination sequence.

Hint: Whenever the guest is started, the script `/hv/bin/observer.sh` is executed which will then wait for the guest to terminate.

Hint: For each KVM guest, a timeout value can be configured in `usr_guest_config.sh`. If the guest does not gracefully terminate it will be forcibly terminated, if the timeout elapsed.

```
$ cd /hv/guests/guestxxxx
$ mousepad usr_guest_config.sh
```

Adjust or add the following line:

```
export shutdown_timeout=900
```

9.5.1 Manual power-off/reboot

You can initiate rebooting or powering off the System by means of the Hypervisor host. For example running the following commands:

```
$ sudo shutdown -h 0
$ # or
$ sudo reboot
```

This will implicitely call the `hv_shutdown` command.

You can also directly call this command to power off the system:

```
$ hv_shutdown
$ # or
$ hv_shutdown --reboot
```

Alternatively, you can use the Hypervisor GUI to shutdown or reboot the system. This will also implicitely call the `hv_shutdown` command.

9.5.2 Reboot/shutdown initiated by Windows KVM guests

In this section we will show how the whole system can be rebooted or shutdown from within a Windows guest:

- Start the Windows guest
- Open the Explorer
- Switch to folder `\\10.0.2.4\qemu\files\WinTools\DesktopShortcuts`
- Copy all the shortcuts onto your desktop (you may have to adjust the shortcuts based on your Windows version and language)
- Run the *"System Shutdown"* or *"System Reboot"* shortcut

Hint: To avoid being asked to allow execution, you may adjust the related Windows settings.

Open the Control Panel, select Internet Options, select security, select local intranet, select sites, select advanced and add `\\10.0.2.4\qemu`

9.5.3 Reboot/shutdown initiated by Linux KVM guests

There are multiple scripts available to either reboot or shut down the Linux guest or the entire system from within a Linux guest itself.

- `hv_linux_reboot:` Reboot the Linux guest
- `hv_linux_shutdown:` Shutdown the Linux guest
- `hv_system_reboot:` Reboot the Hypervisor Host
- `hv_system_shutdown:` Shutdown the Hypervisor Host

Hint:

The scripts for system reboot or shutdown utilize a file share within QEMU.
This file share is established by installing the RTOS Communication Support on the Linux guest system.
You will need `sudo` privileges to use these scripts.
Refer to the `Hypervisor - Ubuntu Guest Guide` chapter `RTOS Communication Support` for installation instructions.

9.5.4 Reboot/shutdown initiated by RTOS guests

To initiate a reboot or shutdown sequence, respective files have to be created in the `/hv/guests/files` folder. Once the shutdown service recognize these files, they will shutdown **all** guests and finally reboot or shutdown the whole system. Within a Real-time Linux guest, this folder is located in `/mnt/rtfiles/files`.

- Initiate system reboot: Create the `/hv/guests/files/system_doreboot` with arbitrary content.

- Initiate system shutdown: Create the `/hv/guests/files/system_doshutdown` with arbitrary content.

9.5.5 Shutdown hooks

The Hypervisor provides a flexible hook mechanism to execute custom scripts at both the **guest** and **system** levels during shutdown or reboot sequences. Hooks are invoked at defined stages (pre- and post-termination) to allow graceful cleanup, logging, or any other custom tasks.

Invocation methods

Shutdown and reboot requests can originate from various sources:

- **Power button** Pressing the PC's power button sends an ACPI event, handled by the `hv_shutdown.service` unit.
- **GUI** Selecting "Shutdown" or "Reboot" in the Hypervisor Host desktop also calls the same `hv_shutdown` logic.
- **Command line**

```
# Standard Linux commands
sudo shutdown -h now
sudo reboot

# Direct hypervisor script
hv_shutdown
hv_shutdown --reboot
```

- **Guest request** A Real-time or KVM guest can request a host shutdown or reboot by creating one of the following trigger files in `/hv/guests/files`:
  - `/hv/guests/files/system_doshutdown`
  - `/hv/guests/files/system_doreboot`

  The shutdown service **watches** for these files and will terminate all guests before powering off or rebooting.

Guest-level hooks

When an individual guest (KVM or RTOS) terminates, it can run a per-guest hook script `vm_shutdown_hook.sh` located in the guest's directory. This script is called **twice**:

```
# Before guest process exits
vm_shutdown_hook.sh --pre_termination <shutdown_flag> <reboot_flag>

# After guest has fully stopped
vm_shutdown_hook.sh --post_termination <shutdown_flag> <reboot_flag>
```

- `<shutdown_flag>` is `1` if the **host** is shutting down.
- `<reboot_flag>` is `1` if the **host** is rebooting.

Use **pre-termination** to quiesce in-guest services or save state, and **post-termination** to collect logs or free resources. You can modify or replace `vm_shutdown_hook.sh` in each guest folder to suit your cleanup needs.

System-level hooks

For host-wide shutdown or reboot, the script `system_init_shutdown_hook.sh` in `$HV_BIN` is invoked at these points:

1. **Initialization (system boot)**

```
system_init_shutdown_hook.sh --start_system
```

2. **shutdown/reboot/terminate**

```
system_init_shutdown_hook.sh --shutdown
```

Use these hooks to restore network interfaces, archive logs, notify external systems, or perform any host-level cleanup before and after the entire shutdown/reboot sequence.

10 RTOS Devices (Partitioning)

10.1 Overview

By default, all devices are assigned for *non* Real-time guests. To assign a specific device to Real-time guests, the RTOSVisor has to be configured appropriately. For each device, a device assignment script has to be created which assigns such a device to RTOS guests in general. In addition, a device configuration file for RTOS guests must be created which assigns such a device to a specific RTOS guest.

Caution: The current script files only support device assignment to a single RTOS guest.

10.2 PCI/PCIe Devices

Hint: This section is valid for all kinds of PCI/PCIe devices. For assigning PCI/PCIe Ethernet devices you should follow the steps described in *PCI/PCIe Ethernet Devices*

10.2.1 Detect pci bus info

The `lspci` command will present a list of all available pci devices. The first number sequence (e.g. `00:00.0`) is the main part of the desired `pci info`. The number sequence is the `vendor:device.function` information. An additonal `-D` parameter shows the domain number.

```
$ sudo lspci -k
```

Here an example output:

```
hvuser@HV-TP106:~$ sudo lspci -k
00:00.0 Host bridge: Intel Corporation Xeon E3-1200 Processor Family DRAM␣
↪Controller (rev 09)
        DeviceName:  Realtek 8211
        Subsystem: Fujitsu Technology Solutions Xeon E3-1200 Processor␣
↪Family DRAM Controller
        Kernel driver in use: ie31200_edac
        Kernel modules: ie31200_edac
00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit Network␣
↪Connection (Lewisville) (rev 04)
        Subsystem: Fujitsu Technology Solutions 82579LM Gigabit Network␣
↪Connection (Lewisville)
        Kernel driver in use: e1000e
        Kernel modules: e1000e
00:1a.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset␣
↪Family USB Enhanced Host Controller #2 (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family USB Enhanced Host Controller
        Kernel driver in use: ehci-pci
00:1c.0 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family␣
↪PCI Express Root Port 1 (rev b4)
```

(continues on next page)

(continued from previous page)

```
        Kernel driver in use: pcieport
00:1c.6 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family␣
↪PCI Express Root Port 7 (rev b4)
        Kernel driver in use: pcieport
00:1c.7 PCI bridge: Intel Corporation 6 Series/C200 Series Chipset Family␣
↪PCI Express Root Port 8 (rev b4)
        Kernel driver in use: pcieport
00:1d.0 USB controller: Intel Corporation 6 Series/C200 Series Chipset␣
↪Family USB Enhanced Host Controller #1 (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family USB Enhanced Host Controller
        Kernel driver in use: ehci-pci
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev a4)
00:1f.0 ISA bridge: Intel Corporation C202 Chipset LPC Controller (rev 04)
        Subsystem: Fujitsu Technology Solutions C202 Chipset LPC Controller
        Kernel driver in use: lpc_ich
        Kernel modules: lpc_ich
00:1f.2 SATA controller: Intel Corporation 6 Series/C200 Series Chipset␣
↪Family 6 port Desktop SATA AHCI Controller (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family 6 port Desktop SATA AHCI Controller
        Kernel driver in use: ahci
        Kernel modules: ahci
00:1f.3 SMBus: Intel Corporation 6 Series/C200 Series Chipset Family SMBus␣
↪Controller (rev 04)
        Subsystem: Fujitsu Technology Solutions 6 Series/C200 Series␣
↪Chipset Family SMBus Controller
        Kernel driver in use: i801_smbus
        Kernel modules: i2c_i801
02:00.0 Ethernet controller: Intel Corporation 82574L Gigabit Network␣
↪Connection
        Subsystem: Fujitsu Technology Solutions 82574L Gigabit Network␣
↪Connection
        Kernel driver in use: e1000e
        Kernel modules: e1000e
03:00.0 VGA compatible controller: Matrox Electronics Systems Ltd. MGA␣
↪G200e [Pilot] ServerEngines (SEP1) (rev 05)
        Subsystem: Fujitsu Technology Solutions MGA G200e [Pilot]␣
↪ServerEngines (SEP1)
        Kernel driver in use: mgag200
        Kernel modules: mgag200
```

Example output with additional `-D` parameter:

```
hvuser@HV-TP106:~$ sudo lspci -kD
0000:00:00.0 Host bridge: Intel Corporation Xeon E3-1200 Processor Family␣
↪DRAM Controller (rev 09)
        DeviceName:  Realtek 8211
        Subsystem: Fujitsu Technology Solutions Xeon E3-1200 Processor␣
↪Family DRAM Controller
        Kernel driver in use: ie31200_edac
        Kernel modules: ie31200_edac
0000:00:19.0 Ethernet controller: Intel Corporation 82579LM Gigabit␣
↪Network Connection (Lewisville) (rev 04)
        Subsystem: Fujitsu Technology Solutions 82579LM Gigabit Network␣
↪Connection (Lewisville)
```

(continues on next page)

(continued from previous page)

```
        Kernel driver in use: e1000e
        Kernel modules: e1000e
...
```

10.2.2 Case 1: PCIe Devices

In case a *non-ethernet PCI Express* (PCIe) device should be assigned to the real-time part, the supplied `hv_addpcidev` command should be used. The device `<bus info>` identified must be used as the first parameter of the `hv_addpcidev` command. The second (**mandatory**) parameter is a unique device name used on the Real-time side. The third (**mandatory**) parameter is a unique number (to be increased sequentially) for each assigned device, starting with `1`.

```
$ hv_addpcidev 0000:02:00.0 rtos_pcidev1 1
```

This call creates two files: `/hv/config/rtos_pcidev1.sh` and `/hv/config/rtos_pcidev1.config`.

In case additional devices shall be assigned:

```
$ hv_addpcidev [DETECTED_BUS_INFO] rtos_pcidev2 2
```

Hint: The RTOS device name is **mandatory**. This parameter **must** be a **unique** name that is used to identify the device. This name will also be used in `filenames` that are created by the `hv_addpcidev` command. In this tutorial the *default* name used is `rtos_pcidev1`.

Caution: If devices with **same** name are assigned to the **same** Rtos the names of the keys in `rtos_pcidev1.sh`, `rtos_pcidev1.config`, etc. **must** be **altered**!

10.2.3 Case 2: Legacy PCI Devices

In case a *non-ethernet Legacy* PCI device should be assigned to the real-time part, the method described in this section has to be applied (all other information described for *non-ethernet* PCIe devices are valid, though). Compared with *non-ethernet* PCIe devices, a fourth parameter of the `hv_addpcidev` command is required.

```
$ sudo hv_addpcidev [DETECTED_BUS_INFO] rtos_pcidev1 1 <interrupt type>
```

Possible values of `<interrupt type>` are **legacy** or **no_interrupt**. If this parameter is **not** provided, the default `MSI interrupt` is used which is **not** feasible for Legacy PCI devices.

10.3 PCI/PCIe Ethernet Devices

Hint: All commands to be executed in the following guide have to be input via the shell. To open the shell right click on the desktop and select *'Open Terminal here'* **or** press `CRTL` + `ALT` + `T`.

10.3.1 Device Identification

In a first step, it is required to determine the Ethernet device that shall be used by the Real-time guest. There are several ways how to detect the desired adapter.

Identify by hardware information

An easy way to identify an adapter is its hardware information:

```
$ lshw -class network
```

returns

```
*-network:1
     description: Ethernet interface
     product: 82545EM Gigabit Ethernet Controller (Copper)
     vendor: Intel Corporation
     physical id: 6
     bus info: pci@0000:02:06.0
     logical name: enp2s0
     version: 01
     serial: 00:0c:29:94:bb:c3
     size: 1Gbit/s
     capacity: 1Gbit/s
     width: 64 bits
     clock: 66MHz
     capabilities: bus_master cap_list rom ethernet physical logical tp␣
↪10bt 10bt-fd 100bt 100bt-fd 1000bt-fd autonegotiation
     configuration: autonegotiation=on broadcast=yes driver=e1000␣
↪driverversion=5.15.0-88-acontis duplex=full ip=172.17.10.26 latency=0␣
↪link=yes mingnt=255 multicast=yes port=twisted pair speed=1Gbit/s
     resources: irq:16 memory:fd580000-fd59ffff memory:fdfe0000-fdfeffff␣
↪ioport:2080(size=64) memory:fd520000-fd52ffff
```

We can see much information helping on identification: The network adapter `enp2s0` is an `'Intel'` type `'82545EM'` with MAC-ID `'00:0c:29:94:bb:c3'` and current link state `'link=yes'`.

Hint: If `RT-Linux` will be used as RTOS, then remember the currently used driver (in this example: `driver=e1000`), as *this* driver can **also** be used in `RT-Linux` with the *attached* device. This could be achieved by a `modprobe [driver_name]` call in the rtos console window.

Identify by link-status change

When working with identical devices, you can identify the desired device by plugging and unplugging the corresponding network cable and observing which device changes its status.

First unplug the cable of the desired Ethernet port and then list the current states:

```
$ ifconfig -a
```

```
enp2s0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 00:0c:29:94:bb:c3  txqueuelen 1000  (Ethernet)
    RX packets 6063  bytes 705772 (705.7 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 292  bytes 62080 (62.0 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now you plug in the cable to the port (to a live network) and list the states again.

```
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.10.26  netmask 255.255.0.0  broadcast 172.17.255.255
    inet6 fe80::28d1:40ed:dd73:b97a  prefixlen 64  scopeid 0x20<link>
    ether 00:0c:29:94:bb:c3  txqueuelen 1000  (Ethernet)
    RX packets 6125  bytes 717920 (717.9 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 350  bytes 71344 (71.3 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Both outputs are similar except the wanted adapter's `<link>` state has changed.

In this example the name of the network adapter is `enp2s0`. It can also be used to identify an adapter by its MAC-ID - in this case `00:0c:29:94:bb:c3`.

10.3.2 Case 1: PCIe Ethernet Devices

In case a *PCI Express* (PCIe) device is assigned, the method described in this section has to be applied. The device name identified above must be used as the first parameter of the `hv_addeth` command. The second (**mandatory**) parameter is a unique device name used on the Real-time side. The third (**mandatory**) parameter is a unique number (to be increased sequentially) for each assigned device, starting with `1`.

```
$ hv_addeth enp2s0 rtos_eth1 1
```

This call creates two files: `/hv/config/rtos_eth1.sh` and `/hv/config/rtos_eth1.config`.

In case additional network devices shall be assigned:

```
$ hv_addeth [DETECTED_NAME] rtos_eth2 2
```

Hint: The RTOS device name is **mandatory**. This parameter **must** be a **unique** name that is used to identify the device. This name will also be used in `filenames` that are created by the `hv_addeth` command. In this tutorial the *default* name used is `rtos_eth1`.

Caution: If devices with **same** name are assigned to the **same** Rtos the names of the keys in `rtos_eth1.sh`, `rtos_eth1.config`, etc. **must** be altered!

10.3.3 Case 2: Legacy PCI Ethernet Devices

In case a *Legacy* PCI device is assigned, the method described in this section has to be applied (all other information described for PCIe devices are valid, though). Compared with PCIe devices, a fourth parameter of the `hv_addeth` command is required.

```
$ sudo hv_addeth [DETECTED_NAME] rtos_eth 1 <interrupt type>
```

Possible values of `<interrupt type>` are **legacy** or **no_interrupt**. If this parameter is **not** provided, the default `MSI interrupt` is used which is **not** feasible for Legacy PCI devices.

10.4 Legacy Serial (COM) Devices (non PCI)

The first step is to get information about COM ports.

```
$ sudo dmesg | grep tty
ttyS0 at I/O 0x3f8 (irq = 4, base_baud = 115200) is a 16550A
```

Write it down somewhere: IRQ 4, IO PORT 0x3F8

The next step is to deactivate the COM port on the host side.

Adjust the files `/etc/grub.d/40_custom` and `/etc/grub.d/41_custom`.
You need to edit the linux kernel boot line which includes the text “linux /boot/vmlinuz- ” and add the following parameters to the end:

```
8250.nr_uarts=0
```

Save the file(s), update the grub menu and reboot.

```
$ sudo update-grub
$ sudo reboot
```

Now add the hardware information about the serial port to RTOSVisor. Create a new file with the name like `/hv/config/rtos_com1.config` and add the following lines:

```
RtosConfig
;-------------------------------------------------------------------------
; COM1 port configuration
;-------------------------------------------------------------------------
[Devices\Rtos\RTOS Serial port (COM1)]
"Type"=dword:0
"PciBus"=dword:0
```

(continues on next page)

(continued from previous page)

```
[Devices\Rtos\RTOS Serial port (COM1)\Interrupt\1]
"Type"=dword:1
"Flags"=dword:3
"Id"=dword:4

[Devices\Rtos\RTOS Serial port (COM1)\IoPort\1]
"Address"=hex:F8,03,00,00,00,00,00,00
"Length"=dword:8
"Flags"=dword:11

;-----------------------------------------------------------------------
; End of file
;-----------------------------------------------------------------------
```

This configuration is valid for one Real-Time OS in your RTOSVisor. Please change the `Devices\RtosX` name if needed.
Please set `Id` to the interrupt number and `Address` is the IO PORT address you previously wrote down.

Edit the file `usr.config` and add a reference to the newly generated file at the end:

```
include "/hv/config/rtos_com1.config"
```

Now start the RT-Linux guest and take a look at the RT-Linux log. Find a line looking like

```
serial8250: ttyS0 at I/O 0x3f8 (irq = 4, base_baud = 115200) is a 16550A
```

If you see this line, it means RT-Linux has recognized your COM port and uses it. The device name for this case is `ttyS0`.

You can test COM port with the command:

```
$ echo "Hello, other side!" > /dev/ttyS0
```

Hint:

If your host computer has several serial ports, but you need to assign only one to RT-Linux, please use the last port.
For example, if your computer has port COM1 and COM2 and you want to use only one port in RT-Linux, change the grub kernel line accordingly: `8250.nr_uarts=1` and use COM2 for RT-Linux.

10.5 RTOS assignment (general)

To assign a specific device to Real-time guests in general, the partitioning script `/hv/config/usr_hvpart.sh` must be adjusted.

The device assignment scripts `<RTOS device name>.sh` usually shall be executed **automatically** on system startup. To accomplish this, you need to add the respective `<RTOS device name>.sh` calls into the file `/hv/config/usr_hvpart.sh`. In our tutorial we use `rtos_eth1` as the unique `<RTOS device name>`, as mentioned earlier.

```
$ mousepad /hv/config/usr_hvpart.sh
```

The `usr_hvpart.sh` file should contain at least the following string **after** editing: `source $HV_CONFIG/rtos_eth1.sh $cmd`

The example below shows how the device with the *unique* name `rtos_eth1` is assigned.

```
#!/bin/bash

cmd="add"
[ $1 == "delete" ] && cmd="delete"

# unbind devices (assign to RTOS)
source $HV_CONFIG/rtos_eth1.sh $cmd
```

Please run the `hv_hvpart` command with the parameter `add` or reboot the system to make the change effective.

```
$ hv_hvpart add
```

You may use the `delete` parameter to assign all RTOS devices back to the Hypervisor Host.

```
$ hv_hvpart delete
```

Hint: The `$HV_BIN/hvpart.sh` script will be automatically started via the *systemd* service controlled via `/etc/systemd/system/hv_part.service`. This script will call the `usr_hvpart.sh` script which includes user specific partitioning commands. This service can be *enabled* or *disabled* as shown below (by default, it is enabled)

```
$ sudo systemctl enable /hv/services/hv_part.service
$ sudo systemctl disable hv_part
```

Caution: The System Manager configuration tool will write the device assignment into `/hv/bin/hvpart.sh`, you must **not** change this file. Effectively, all devices listed in `/hv/bin/hvpart.sh` and `/hv/config/usr_hvpart.sh` will be assigned to the RTOS guests.

10.5.1 Verification

de-assignment check using Hypervisor Host

In a first step, check if the Hypervisor Host's *original* driver is **not** used in conjunction with the devices assigned to the Real-time guest.

```
$ lspci -k
```

The output will look similar like the following excerpt:

```
    :          :        :          :        :          :        :          :
↪      :          :
    :          :        :          :        :          :        :          :
↪      :          :
    :          :        :          :        :          :        :          :
↪      :          :
01:00.0 Ethernet controller: Intel Corporation I210 Gigabit Network
↪Connection (rev 03)
        Subsystem: Intel Corporation I210 Gigabit Network Connection
        Kernel driver in use: igb
        Kernel modules: igb
02:00.0 Ethernet controller: Intel Corporation I210 Gigabit Backplane
↪Connection (rev 03)
        Subsystem: Intel Corporation I210 Gigabit Backplane Connection
        Kernel driver in use: pci-stub
        Kernel modules: igb
03:00.0 Ethernet controller: Intel Corporation I210 Gigabit Backplane
↪Connection (rev 03)
        Subsystem: Intel Corporation I210 Gigabit Backplane Connection
        Kernel driver in use: pci-stub
        Kernel modules: igb
    :          :        :          :        :          :        :          :
↪      :          :
    :          :        :          :        :          :        :          :
↪      :          :
    :          :        :          :        :          :        :          :
↪      :          :
```

In the above example, the instance `01:00.0` is used by Ubuntu (driver: *igb*, *e1000e* etc.) and the instances `02:00.0` and `03:00.0` are assigned to a Real-time guest (driver: *pci-stub*).

10.6 Assign a device to a specific Real-time guest

After creating the device configuration file `<RTOS device name>.config`, it needs to be included into the guest configuration file to become effective for the respective guest. The next steps describe how to add this device to the example Real-time Linux guest, we use `rtos_eth1` as the unique `<RTOS device name>`.

Hint: Initially, the Hypervisor Host does not provide any example guest folders. To switch to an example guest, you must execute the corresponding initialization. For instructions on how to initialize the examples, refer to *Example guest folders*.

```
$ hv_open_example rt-linux
$ hv_sync_example rt-linux
$ cd /hv/guests/guestrtlinux
$ mousepad ./usr.config
```

Add the corresponding `rtos_eth1.config` entry to the `includes` section of the configuration file.

The following example shows the 'modified' `usr.config` file:

```
RtosConfig
;-------------------------------------------------------------------------
; acontis technologies GmbH
;
; Guest user configuration
;-------------------------------------------------------------------------

#include "/hv/config/rtos_eth1.config"

;-------------------------------------------------------------------------
; End of file
;-------------------------------------------------------------------------
```

10.6.1 Real-time guest assignment check

Real-time Linux

After assigning a device to the RT-Linux a driver could be loaded for that device.

Run the rtos:

```
$ cd /hv/guests/guestrtlinux
$ hv_guest_start -view
```

Log in into Real-Time Linux and run `modprobe [driver_name]`:

```
$ vmf64 login: root
$ password:  root
$ modprobe e1000
```

Hint: In the default case the *previously* used Hypervisor Host driver can **still** be used in RT-Linux.

RTOS-32

We use the `RTOS-32Demo` application to quickly verify, if the assigned netwok adapter is visible to the rtos part. First, the rtos32 example must be initialized and the `usr.config` file needs to be adjusted.

```
$ hv_open_example rtos-32
$ hv_sync_example rtos-32
$ cd /hv/guests/guestrtos32
$ mousepad ./usr.config
```

Add the corresponding `rtos_eth1.config` entry to the `includes` section of the configuration file.

The following example shows the 'modified' `usr.config` file:

```
RtosConfig
;-------------------------------------------------------------------------
; acontis technologies GmbH
;
; Guest user configuration
;-------------------------------------------------------------------------

#include "/hv/config/rtos_eth1.config"

;-------------------------------------------------------------------------
; End of file
;-------------------------------------------------------------------------
```

Adjust the guest configuration setting to prepare starting the *RTOS-32Demo*:

```
$ cd /hv/guests/guestrtos32
$ mousepad usr_guest_config.sh
```

and add the following line:

```
export osImage=$HV_ROOT/guests/guestrtos32/Loader.bin
```

Adjust the link to the demo application:

```
$ cd /hv/guests/guestrtos32
$ rm rtos32app.dlm
$ ln -s /hv/guests/guestrtos32/files/RTOS-32Demo.dlm rtos32app.dlm
```

Run the demo:

```
$ cd /hv/guests/guestrtos32
$ hv_guest_start -view
```

The output should look like this for the used network adapter (`8086:1533`) in this tutorial:

```
PCI BIOS Information

Vendor Device Bus Dev Func Class  Int IRQ TLat MSI Type
-------------------------------------------------------
FFFE   FFFE     0   0    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0   2    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  20    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  22    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  26    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  27    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  28    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  29    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  30    0  000000  -    0    0  -  Invalid class code
FFFE   FFFE     0  31    0  000000  -    0    0  -  Invalid class code
8086   1533     1   0    0  020000  A    5    0  X  Network controller
FFFE   FFFE     2   0    0  000000  -    0    0  -  Invalid class code
10EC   8169     3   5    0  020000  A    4   64  -  Network controller
Current Date/time: 01/01/2018 00:00:00
Current Date/time: 01/01/2018 00:00:01
Current Date/time: 01/01/2018 00:00:02
Current Date/time: 01/01/2018 00:00:03
Current Date/time: 01/01/2018 00:00:04
```

(continues on next page)

(continued from previous page)

```
List of threads:
Name             Prio State      R.Del FStck  MStck Scheds  CTime  CT%
-------------------------------------------------------------------------
Main Task           5 Current          63060  61276     11  -      -
Idle Task           0 Ready             2164   2164    526  -      -
Comm                5 Delaying       1 32832  31724    527  -      -
IPTASK              6 BlckdWait         4128   3940      2  -      -
IPTIMER             6 Delaying      11  4120   3940     44  -      -

Interrupts:
 IRQ    Calls  FreeStack  Doubles  Time
-----------------------------------------
   0      526        744        0  -

Network:
 Ping will respond at IP address 192.168.157.2
```

In the above example, one `8086:1533` device at `BDF 1,0,0` is assigned to the Real-time guest.

Finally, terminate the console connection to the real-time guest and stop the Real-time guest OS:

`CTRL + C`

```
$ cd /hv/guests/guestrtos32
$ hv_guest_stop
```

10.7 RTOS de-assignment

To de-assign a specific device from Real-time guests, the partitioning script `/hv/config/usr_hvpart.sh` must be adjusted.

In a first step, please de-assign **all** devices from RTOS guests using the `delete` parameter of the hv_hvpart command (assign all RTOS devices back to the Hypervisor Host):

```
$ hv_hvpart delete
```

Then you need to remove or uncomment the respective `<RTOS device name>.sh` calls from the file `/hv/config/usr_hvpart.sh`. Here we use `rtos_eth1` as the unique `<RTOS device name>`, as mentioned earlier.

```
$ mousepad /hv/config/usr_hvpart.sh
```

The `usr_hvpart.sh` file should contain at least the following string **after** editing: `source $HV_CONFIG/rtos_eth1.sh $cmd`

The example below shows how the device with the *unique* name `rtos_eth1` is assigned.

```
#!/bin/bash

cmd="add"
[ $1 == "delete" ] && cmd="delete"

# unbind devices (assign to RTOS)
# source $HV_CONFIG/rtos_eth1.sh $cmd       --> uncomment or remove this␣
↪line
```

Finally, run the `hv_hvpart` command with the parameter `add` or reboot the system to assign the remaining devices to RTOS guests again.

```
$ hv_hvpart add
```

11 Hypervisor Host Network

Note: This section introduces the networking capabilities of the Hypervisor Host. You will learn how the Host manages the internal virtual network, configures forwarding or bridging for external access, and adjusts the interface settings for different network scenarios. By following the steps below, you can fine-tune how the Hypervisor Host and its guests communicate both internally and with external networks.

11.1 Virtual network

The hypervisor provides a virtual network. The Hypervisor Host and all the guests can be connected to this virtual network.

The IP addresses are set to fixed values.

The default IP addresses are:

Hypervisor Host: `192.168.157.1`

First RTOS: `192.168.157.2`

Windows or Linux example guest: `192.168.157.3`

Hint: The Hypervisor Host virtual network IP address is initially set when installing the Hypervisor (via calling the `/hv/bin/inithv.sh` script). For more details, see *Internal virtual network*.

11.2 Network Forwarding from external computer to the RTOS

11.2.1 Hypervisor Host preparation

If the RTOS (or any other OS connected to the virtual network) shall be accessed via TCP/IP from a **single** external system, traffic can be forwarded to the virtual network. Execute the following steps to forward traffic from a specific external computer to the RTOS:

- enable temporary network forwarding in the Hypervisor Host:

  ```
  $ sudo sysctl -w net.ipv4.ip_forward=1
  ```

- or, alternatively, enable permanent network forwarding in the Hypervisor Host:

  ```
  $ sudo mousepad /etc/sysctl.conf
  $
  $ uncomment the following line:
  $   net.ipv4.ip_forward=1
  ```

- determine the IP address of the Hypervisor Host. You can use the `ifconfig` command to accomplish this.

Caution:

Assure the Default Gateway in the RTOS is set to the Hypervisor Host virtual network IP address (`192.168.157.1`)!
For RT-Linux it is set by default. For other RTOS you need to check the RTOS documentation how to accomplish this.

IPTables Setup

In case you have multiple network adapters connected to the Hypervisor Host or installed Docker in the Hypervisor host, you may have to create iptable rules for correct IP forwarding. This section provides steps to configure `iptables` rules on the Hypervisor Host and ensure they are persistent across reboots by using a custom script and a systemd service.

Step 1: Create a Script to Apply the Rules

Below we assume the network device `ens33` is connected to the outside network and shall be used for IP forwarding. You may have to adjust this device according to your setup.

1. Create a script file to store the `iptables` rules:

```
sudo mkdir /etc/iptables
sudo nano /etc/iptables/rules.sh
```

2. Add the `iptables` rules to this script. Example content:

```
#!/bin/sh
iptables -A FORWARD -i ens33 -o vnet0 -j ACCEPT
iptables -A FORWARD -i vnet0 -o ens33 -j ACCEPT
iptables -t nat -A POSTROUTING -o ens33 -j MASQUERADE
```

3. Make the script executable:

```
sudo chmod +x /etc/iptables/rules.sh
```

Step 2: Create a systemd Service to Run the Script on Boot

1. Create a systemd service file:

```
sudo nano /etc/systemd/system/iptables-rules.service
```

2. Add the following content to the service file:

```
[Unit]
Description=Load iptables rules

[Service]
Type=oneshot
ExecStart=/etc/iptables/rules.sh
RemainAfterExit=yes
```

(continues on next page)

(continued from previous page)

```
[Install]
WantedBy=multi-user.target
```

3. Enable the systemd service to ensure it runs on boot:

```
sudo systemctl enable iptables-rules.service
```

4. Start the systemd service to make if effective:

```
sudo systemctl start iptables-rules.service
```

Verification

After completing the setup, you can verify that the rules are applied correctly by checking the `iptables` configuration and the status of the systemd service.

1. Check the `iptables` rules:

```
sudo iptables -L -v -n
```

2. Check the status of the systemd service:

```
sudo systemctl status iptables-rules.service
```

By following these steps, you ensure that your `iptables` rules are persistent and automatically applied whenever your system reboots.

11.2.2 Forwarding from external Windows computer

- open a Command Window with Administrator rights on your Windows PC
- run the following command (replace `AAA.BBB.CCC.DDD` with the appropriate IP address of the Hypervisor Host):

```
route add 192.168.157.0 mask 255.255.255.0 AAA.BBB.CCC.DDD
```

11.2.3 Forwarding from external Linux computer

- open a Terminal Window on your Linux PC
- run the following command (replace `AAA.BBB.CCC.DDD` with the appropriate IP address of the Hypervisor Host):

```
ip route add 192.168.157.0/24 via AAA.BBB.CCC.DDD
```

11.3 Bridge virtual and physical network

If the RTOS (or any other OS connected to the virtual network) shall be accessed via TCP/IP from **any** external system, the virtual network and the respective physical network have to be bridged.

In the folder `/hv/config` you can find the template configuration file `brvnetconfig.sh` for the bridge configuration. Note, the IP address of the virtual network inside the RTOS guest need to be adjusted appropriately, see below for more details.

11.3.1 Bridge configuration

First step: determine, which network adapter should be bridged. Search for `<link>` entry and get the adapter name.

```
$ ifconfig -a
```

In this case it's `enp2s0`. The current `$IP$` address of `enp2s0` is `inet 172.17.10.53` and the network mask is `255.255.0.0`.

```
rtv@rtv-TEST:~$ ifconfig -a
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 172.17.10.53  netmask 255.255.0.0  broadcast 172.17.255.255
    inet6 2a02:590:801:2c00:7170:3747:f835:a1cb  prefixlen 64  scopeid 0x0
↪<global>
    inet6 fe80::fe6f:c5f8:c5cd:e3cd  prefixlen 64  scopeid 0x20<link>
    inet6 2a02:590:801:2c00:96b0:b8a:2c58:6c91  prefixlen 64  scopeid 0x0
↪<global>
    ether 90:1b:0e:18:c9:83  txqueuelen 1000  (Ethernet)
    RX packets 116751  bytes 22127837 (22.1 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 74453  bytes 551331072 (551.3 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

enp3s5: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
    ether 74:ea:3a:81:4b:1d  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop  txqueuelen 1000  (Local Loopback)
    RX packets 201  bytes 14798 (14.7 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 201  bytes 14798 (14.7 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vnet0: flags=99<UP,BROADCAST,NOTRAILERS,RUNNING>  mtu 1500
    inet 192.168.157.1  netmask 255.255.255.0  broadcast 192.168.157.255
    ether 00:60:c8:00:00:00  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 59  bytes 10381 (10.3 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Next step: determine the default gateway.

```
$ ip route ls
```

You will get an output like `default via 172.17.5.2 dev enp2s0 proto dhcp metric 100`.

Next step: determine the DNS server.

```
$ resolvectl status | grep "Current DNS Server"
```

You will get an output like `Current DNS Server: 172.17.5.9`.

Next step: Adjust `brvnetconfig.sh` with the detected values of `ifconfig`:

```
$ mousepad /hv/config/brvnetconfig.sh
```

Values:

- `netif="enp2s0"`
- `defaultgw="172.17.5.2"`
- `dns="172.17.5.9"`
- `vnetbrip="172.17.10.53"`
- `vnetbrnm="255.255.0.0"`
- `#vnetbrmac=` comment in and adjust value only if there are collisions with **'same'** `MAC-IDs` on the network.

```
#!/bin/bash

# Ethernet network interface to bridge with VM.
# ethernet interface to bridge with vnet
netif="enp2s0"

# Default gateway
# How to determine the default gateway:
#       Use the command ip route ls
#           default via 172.17.5.2 dev enp2s0 proto dhcp metric 100
#           172.17.0.0/16 dev enp2s0 proto kernel scope link src 172.17.
↪10.4 metric 100
#       The default gateway here is "172.17.5.2"
defaultgw="172.17.5.2"        # default gateway

# DNS server
# How to determine the default gateway:
#       Use the following command: resolvectl status | grep "Current DNS␣
↪Server"
#            Current DNS Server: 172.17.5.9
#       The DNS server here is "172.17.5.9"
dns="172.17.5.9"

# Bridge settings
# The bridge replaces the former network device used by the hypervisor to␣
↪connect to the network.
# See above results provided by the ifconfig -a command
#       enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
#               inet 172.17.10.53  netmask 255.255.0.0  broadcast 172.17.
↪255.255
```

(continues on next page)

(continued from previous page)

```
#       In this example, the bridge IP address is 172.17.10.53 and the␣
↪network mask is 255.255.0.0
vnetbrip="172.17.10.53"
vnetbrnm="255.255.0.0"

# the values below are default values, typically not to be changed
vnetip="192.168.157.1"
vnetnm="255.255.255.0"
#vnetbrmac="54:52:00:ac:30:10       # by default, the MAC address of the␣
↪physical network is used
vnet="vnet0"
vnetbr="vnetbr"
```

RT-Linux Guest IP address settings

If the network is bridged, the IP address of RT-Linux must be adjusted properly. The settings are stored in `/hv/guests/guestrtlinux/guest.config`. The IpAddress has to be set to a unique address in your company network, assure the entries are uncommented! The MacAddress has to be adjusted to a unique value only if more than one RT-Linux guest is bridged, in that case, please adjust the last value from `12` to `13`, `14` etc.

```
; This must be set correctly if the vnet device is bridged in the␣
↪Hypervisor Host
[Rtos\Vnet\0]
    "IpAddress"="172.17.10.239"
    "MacAddress"="AA:BB:CC:DD:E0:12"
```

Bridge activation

- After configuring the bridge parameters, you can create the bridge:
  - First, start the RTOS to assure the virtual network is available.
  - Run the `hv_brvnetset` command.
- Remove the bridge - Run the `hv_brvnetclr` command.

11.4 Hypervisor Host network configuration

The Hypervisor Host network can be configured using automatic IP address configuration (DHCP), manual IP address configuration or disabled network. To simplify the process, the `hv_netconf` command is provided.

Caution:

The Hypervisor Host network configuration has to match with the KVM guest network settings (e.g. Windows or Ubuntu guest)

One of the guest network settings is determined by parameter `netif_mode` in the guest configuration file (e.g. `usr_guest_config.sh` located in the `GUEST_FOLDER`).

If the Hypervisor Host and guest settings do not match, the behaviour is undefined.

11.4.1 Automatic network configuration

This is the default mode. Nothing has to be changed if this mode shall be used. In case the networking had been adjusted manually, you can switch back to the automatic configuration as follows.

```
$ hv_netconf -auto
```

A single network interface can be set into automatic mode as follows.

```
$ hv_netconf %DEVICE% -auto
```

For example:

```
$ hv_netconf enp1s0 -auto
```

11.4.2 Manual network configuration

If you want the Hypervisor Host to be configured manually, you need to adjust the settings accordingly.

```
$ hv_netconf %DEVICE% -manual IP-address netmask-bits gateway-IP dns-IP
```

For example:

```
$ hv_netconf enp1s0 -manual 192.168.178.188 24 192.168.178.1 8.8.8.8
```

Then, configure the guest.

```
$ cd GUEST_FOLDER
$ mousepad usr_guest_config.sh
```

Change the respective configuration values.

```
netif_mode=0
netif_m=...
defaultgw_m=...
dnsgw_m=...
brip_m=...
brnm_m=...
```

11.4.3 Disabled network

If you want the Hypervisor Host not to use the network, you need to adjust the settings accordingly. In case the PC currently is connected with the LAN and you want to use this connection for a guest, you need to determine the device name before disabling the Hypervisor Host network. You may use the `ifconfig` command for that purpose.

```
$ sudo ifconfig
```

Next, disable the network of the Hypervisor Host.

```
$ hv_netconf -off
```

You need to turn off IPv6 as follows.

```
$ sudo mousepad /etc/sysctl.conf
```

Insert the following lines at the bottom of this file:

```
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Make this effective:

```
$ sudo sysctl -p
```

Then, configure the guest.

```
$ cd GUEST_FOLDER
$ mousepad usr_guest_config.sh
```

Change the respective configuration values.

```
netif_mode=2
netif_m=...
```

Caution: You must define the network device that shall be used in the guest by setting the parameter `netif_m` to the name you have determined above.

Hint: To re-enable the Hypervisor Host network in automatic mode, run the `hv_netconf` command again:

```
$ hv_netconf -auto
```

And change the `netif_mode` configuration value.

```
$ cd GUEST_FOLDER
$ mousepad usr_guest_config.sh
```

```
netif_mode=1
```

11.4.4 Internal virtual network

The Hypervisor provides an internal virtual network using fixed IP address settings. The Hypervisor Host and all the guests can be connected to this virtual network. All guests also need to use fixed IP address settings. The device name for the virual network is **vnet0**. This device is **not** managed by the NetworkManager and settings **cannot** be changed using the `hv_netconf` command.

To change the settings, please adjust the configuration file `/etc/network/interfaces`. The IP addresses of the RTOS and KVM guests have to be adjusted accordingly.

Hint: If the `vnet0` device would be managed by the NetworkManager, a specific network connection for `vnet0` may be created using a *fixed* IP address. But even in such case, the NetworkManager may try to use one of the default network connections (e.g. Wired Connection 1) which are not bound to a specific device. As a result, the NetworkManager will try to get the IP address for `vnet0` via DHCP. As there is no DHCP server on this network, this will lead to a timeout. In addition, additional (physical) network

devices will also **not** work (or work only after a very long time) unless there is a specific connection defined for such device (using the `hv_netconf` command).

11.5 Hypervisor network service/port management

In many scenarios, you must manage network-related services on the Hypervisor Host — for example, to ensure unnecessary services are disabled (for security reasons) or to enable services like SSH, FTP, and SMB.
Additionally, you may want to see which TCP/UDP ports are currently in use, identify which processes own them, and forcefully terminate any unwanted processes.

11.5.1 Introduction

The following steps outline how you can:

1. **Determine which network services exist** on your system and verify if they are installed.
2. **Enable or disable** specific network services (or all at once) for security or configuration reasons.
3. **Check which network ports** are in use and **identify processes** (e.g., daemons like SSH, FTP, or custom applications) associated with those ports.
4. **Terminate processes** bound to a port, including their child processes, if necessary for security or troubleshooting.

All of these functions are accessible through the `hv_netsvc_mgr` command, which you can run at any time on the Hypervisor Host. It supports multiple command-line options for listing and managing services as well as ports and processes.

11.5.2 Command Reference

Below is a high-level overview of the `hv_netsvc_mgr` command. It can:

- **Enable/disable/show** a predefined list of network services.
- **Show open ports**, filtering by “all,” “open,” or “connected” states.
- **Show or kill** all processes using a particular port (including their child processes).

Usage Examples

```
# Show the status of all service:
hv_netsvc_mgr -svcshow

# Show the status of a specific service:
hv_netsvc_mgr -svcshow ssh

# Enable a single service:
hv_netsvc_mgr -svcena smbd
```

(continues on next page)

(continued from previous page)

```
# Disable all network services:
hv_netsvc_mgr -svcdis all

# Show all open ports:
hv_netsvc_mgr -portshow all

# Show only actively established (ESTAB) connections:
hv_netsvc_mgr -portshow conn

# Show processes associated with port 80:
hv_netsvc_mgr -prcshow 80

# Kill all processes associated with port 22 (e.g., SSH):
hv_netsvc_mgr -prckill 22
```

Notes and Tips

- **Service Management**: By default, `hv_netsvc_mgr` references a built-in array of known network services (e.g., SSH, vsftpd, smbd). If a service is missing from this array, the command won't manage it unless you add it to the array.
- **Sockets**: Some services also have corresponding socket units (e.g., `rpcbind.socket`). These are included so you can enable/disable or show them as needed.
- **Systemd vs. Manual Processes**: For ephemeral ports or user processes, you might not see a direct systemd "service." In that case, the process-oriented commands (`-prcshow` / `-prckill`) help identify the actual PID and any child processes that hold a port.
- **Security Considerations**: Disabling unneeded services or stopping processes on unused ports lowers the **attack surface**. But ensure you do not disable essential services (for instance, SSH if you rely on remote access).

By using `hv_netsvc_mgr`, you have a single tool to audit and control the Hypervisor Host's network services and ports, keeping it both **functional** and **secure**.

11.5.3 Network airgapping

For security reasons, you may want the Hypervisor to stay **completely disconnected** from the externel network. For that purpose the network must be disabled and all network related services should be stopped and disabled. Furthermore, you may also want to terminate all processes that currently have open network connections.

In a first step, disable the network itself as described in *Disabled network*.

Then proceed according to the following steps:

```
# stop and disable the System Manager service
hv_sysmgr stop
hv_sysmgr disable

# stop and disable all network related services
hv_netsvc_mgr -svcdis all
```

(continues on next page)

(continued from previous page)

```
# reboot the system
sudo reboot
```

Instead of rebooting, you may also forcibly terminate all processes with open network connections.

```
hv_netsvc_mgr -prckill all
```

Most likely, an automatic logout wil then occur. You will have to log in again. Verify, if no services are active and no ports are open.

```
hv_netsvc_mgr -svcshow
hv_netsvc_mgr -portshow
```

If there are open ports, check which processes are using them:

```
hv_netsvc_mgr -prcshow
```

Hint: If you disable the Hypervisor host from the network, you need to assure that the mode for the KVM guest networking is set up properly. The default mode will not work in such cases. See *Automatic external network connection (dynamically bridged)*.

11.5.4 Re-enable network access

Follow the below steps to re-enable the network access again.

```
# enable and start all network related services
hv_netsvc_mgr -svcena all

# enable and start the System Manager service
hv_sysmgr enable
hv_sysmgr start
```

Finally, re-enable the network itself as described in *Disabled network*.

12 USB Devices

12.1 USB device access for RT-Linux guests

Caution: Starting with version 9.0, USB device access for RT-Linux guests is no longer supported in RTOSVisor.

A single USB device can be accessed from within one single RT-Linux guest using `usbip`.
This will expose a USB device from the Hypervisor Host to the RT-Linux guest via the virtual network.

12.1.1 Hypervisor Host usbip package

In a first step, the respective software package for the RTOSVisor may have to be installed. Run the following command to check if the package is already available:

```
$ which usbipd
```

The package is available if you get the following result: `/usr/bin/usbipd`.

acontis linux kernel comes with preinstalled `linux-extra-` package, so it already includes `usbip`. If `usbip` package is **not avaialble**, please contact your technical support.

12.1.2 RT-Linux

The standard RT-Linux image shipped with the hypervisor does not support `usbip`. You need to exchange this image by the separately provided `rtlinux515.x64-usbip.bin` image file. This file is part of the CODESYS package and located in `/hv/guests/etc/rt-linux/files/codesys/rtlinux`. You need to initialize the RT-Linux example and adjust the configuration:

```
$ hv_open_example rt-linux
$ hv_sync_example rt-linux
$ cd /hv/guests/guestrtlinux
$ mousepad guest_config.sh
```

Adjust the following line:

```
export osImage=$HV_ROOT/guests/etc/rt-linux/files/codesys/rtlinux/
↪rtlinux515.x64-usbip.bin
```

12.1.3 Prepare USB device exposal

The USB device which shall be exposed to RT-Linux needs to be prepared for exposal. In a first step you will need to determine the vendor and device ID of the USB device you want to expose. Insert the USB device and execute the following commands.

```
$ hv_hvusbip -init
$ hv_hvusbip -list
```

Hint: The call with the `-init` parameter is only required once.

A list of USB devices will be shown.

```
device list
***********
 - busid 3-2 (046d:c52b)
   Logitech, Inc. : Unifying Receiver (046d:c52b)

 - busid 3-3 (046d:c52f)
   Logitech, Inc. : Unifying Receiver (046d:c52f)

 - busid 3-5 (0e8d:0608)
   MediaTek Inc. : unknown product (0e8d:0608)

 - busid 3-8 (064f:2af9)
   WIBU-Systems AG : CmStick (HID, article no. 1001-xx-xxx) (064f:2af9)

 - busid 4-4 (18a5:0243)
   Verbatim, Ltd : Flash Drive (Store'n'Go) (18a5:0243)
```

In this example, we will expose the WIBU CmStick device to the guest.
According to the above output, its vendor device id is **064f:2af9**

Then you need to insert the vendor device ID into the respective configuration file.

```
$ cd /hv/config
$ mousepad usbip_exposed_device.sh
```

Insert the appropriate vendor device ID.
Don't forget to remove the comment at the beginning of the line!

```
# one single device currently can be exposed to the RTOS
export usb_vendev=064f:2af9  # WIBU CodeMeter USB dongle
```

Hint: After booting, the `hv_usbip_expose.service` located in `/hv/services` will prepare USB device exposal.

12.1.4 Expose USB device to the guest

You need to copy the `/hv/config/usbip_exposed_device.sh` into the respective guest folder to expose the USB device.

```
$ cd GUEST_FOLDER
$ cp /hv/config/usbip_exposed_device.sh .
```

Hint:

When starting RT-Linux, the `GUEST_FOLDER/usbip_gen_init.sh` script will dynamically create the `GUEST_FOLDER/files/usbip_init.sh` script.
This sub script will be called when RT-Linux is booted via the `GUEST_FOLDER/files/autostart.sh` script.

Caution: You must not adjust the `GUEST_FOLDER/files/usbip_init.sh` script manually, it will be overwritten when RT-Linux is started!

You may insert specific autostart activities when the RT-Linux guest starts.

```
$ cd GUEST_FOLDER
$ mousepad usbip_gen_init.sh
```

At the very bottom the required autostart activity for a specific WIBU Codemeter USB device is shown. Please adjust the script `GUEST_FOLDER/usbip_gen_init.sh` according to your needs.

```
# insert usbip specific autostart activities here
if [ $usb_vendev == "064f:2af9" ]; then
    # echo "start codemeter daemon"
    echo "start-stop-daemon --start --quiet --chuid root --exec /usr/sbin/
↪CodeMeterLin" >>./rtfiles/usbip_init.sh
    echo "to verify CodeMeter status run 'cmu --cmdust'" >>./rtfiles/usbip_
↪init.sh
fi
```

Then, the `usbip` expose service needs to be enabled which will start the usbip server on the Hypervisor Host:

```
$ sudo systemctl enable /hv/services/hv_usbip_expose.service
```

Finally you should start the service to make your changes effective. You may also reboot to do so.

```
$ sudo systemctl start hv_usbip_expose
```

Hint:

This service will run the script `/hv/config/usbip_expose.sh` which will finally expose the device.
You can run the command `systemctl status hv_usbip_expose.service` to verify if exposing worked correctly.

Caution:

If the USB device is removed and re-inserted, it has to be exposed again.

Furthermore, in RT-Linux the appropriate steps (executed in `usbip_init.sh`) also have to be executed again.

After you have started the service, you can check if the device is correctly exposed - before starting RT-Linux. Connect the USB device and then run the following command.

```
$ usbip list -r 127.0.0.1
```

A list of exportable USB devices will be shown. The one that you have set before should be shown.

```
Exportable USB devices
======================
 - 127.0.0.1
        3-3: WIBU-Systems AG : CmStick (HID, article no. 1001-xx-xxx)␣
↪(064f:2af9)
           : /sys/devices/pci0000:00/0000:00:14.0/usb3/3-3
           : (Defined at Interface level) (00/00/00)
           :  0 - Human Interface Device / No Subclass / None (03/00/00)
```

Finally you can start RT-Linux, the USB device should be visible then. To verify this, you can run the following command (inside RT-Linux!).

```
$ usbip port
```

A list of imported USB devices will be shown. The one that you have set before should be shown.

```
Imported USB devices
====================
Port 00: <Port in Use> at Full Speed(12Mbps)
       WIBU-Systems AG : CmStick (HID, article no. 1001-xx-xxx) (064f:2af9)
       1-1 -> usbip://192.168.157.1:3240/3-3
           -> remote bus/dev 003/007
```

Hint:

To stop exposing USB devices, you have to comment the device in `/hv/config/usbip_exposed_device.sh`.

```
# one single device currently can be exposed to the RTOS
# export usb_vendev=064f:2af9  # WIBU CodeMeter USB dongle
```

Then you should disable the service.

```
$ sudo systemctl disable hv_usbip_expose
```

12.1.5 Hypervisor Host usbip low level services

The `usbip` low level services are used by the scripts and services of the Hypervisor Host.
This background information may be helpful for debugging and diagnosis purposes.

Loading kernel modules:

```
$ sudo modprobe usbip_host
$ sudo modprobe usbip_core
```

Start the usbip daemon:

```
$ sudo usbipd -D
```

Show all USB connected devices:

```
$ usbip list -l
```

Expose a specific USB device:

```
$ sudo usbip bind -b busid
```

Show all currently exposed USB devices that are not in use:

```
$ usbip list -r 127.0.0.1
```

Stop exposing a specific USB device:

```
$ sudo usbip unbind -b busid
```

12.1.6 RT-Linux usbip low level services

The `usbip` low level services are used by the scripts and services of the RT-Linux guest.
This background information may be helpful for debugging and diagnosis purposes.

Loading kernel modules:

```
$ modprobe vhci-hcd
```

Show exposed devices:

```
$ usbip list -r 192.168.157.1
```

Attach exposed device:

```
$ usbip attach -r 192.168.157.1 -b "Device bus ID"
```

Show ports in use by attached devices:

```
$ usbip port
```

Detach a specific port:

```
$ usbip detach -p "port"
```

12.2 Windows/Linux USB guest access (non automatic mode)

In case the USB device is plugged in while the guest is already running, you need to determine some information about the USB device.

Using the `lsusb` command, a list of connected USB devices can be found:

```
lsusb
  Bus 002 Device 002: ID 0b95:1790 ASIX Electronics Corp. AX88179 Gigabit␣
↪Ethernet
  Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
  Bus 001 Device 003: ID 8087:0025 Intel Corp. Wireless-AC 9260 Bluetooth␣
↪Adapter
  Bus 001 Device 006: ID 064f:2af9 WIBU-Systems AG CmStick (HID, article␣
↪no. 1001-xx-xxx)
  Bus 001 Device 005: ID 046d:c050 Logitech, Inc. RX 250 Optical Mouse
  Bus 001 Device 004: ID 046a:0011 Cherry GmbH G83 (RS 6000) Keyboard
  Bus 001 Device 002: ID 1a40:0101 Terminus Technology Inc. Hub
  Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Then you need to run the `hv_guest_monitor` command.

```
$ cd GUEST_FOLDER
$ hv_guest_monitor
```

If you want to dynamically connect the first USB device (Bus 002, Device 002), type in the following command:

```
device_add usb-host,id=MyUsbDevice,hostbus=2,hostaddr=2
```

The id value can be selected freely but it has to be unique in case multiple USB devices are connected.

Hint: More information can be found here:

- https://github.com/qemu/qemu/blob/master/docs/usb2.txt
- https://unix.stackexchange.com/questions/426652/connect-to-running-qemu-instance-with-qemu-monitor/476617
- https://wiki.archlinux.de/title/QEMU#USB_Peripherie

12.3 Windows/Linux USB guest access (passthrough mode) for non Real-time guests

If a specific physical USB port shall be permanently used by the guest, this can be accomplished using USB passthrough mode.
In this case, the hypervisor will monitor a specific USB port and automatically passthrough a USB device (for example a USB 3.0 Stick) to an active guest.

In the first step, you need to determine the USB `hostbus` and `hostport` value pairs for the selected physical USB port.

Caution: Depending on the USB type (USB1/2 or USB3) there will be *different* values even if the same physical port is used.

Caution:

If a USB mouse is connected to a specific USB port, you must not passthrough this USB port unless you are using graphics passthrough mode.

This means, the physical USB port where such mouse is connected must not be used for other devices.

Also, the USB mouse must not be connected to any physical USB port which is passed through to the guest.

If this port is passed through, the mouse pointer will not be visible!

Hint: More information can be found here:

- https://qemu.weilnetz.de/doc/6.0/system/usb.html
- https://qemu-project.gitlab.io/qemu/system/devices/usb.html
- https://www.kraxel.org/blog/2018/08/qemu-usb-tips/

12.3.1 USB1/2 devices

Connect a USB2 stick to the USB port you want to passthrough and execute the following on the Hypervisor Host:

```
$ lsusb -t
```

You will get a result similar to

```
1 /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2 /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
3     |__ Port 4: Dev 18, If 0, Class=Mass Storage, Driver=usb-storage, 480M
4     |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
5     |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB hostbus value for this physical port using USB1/USB2 devices is 1 (see line 2)
The USB hostport value for this physical port using USB1/USB2 devices is 4 (see line 3)

12.3.2 USB3 devices

Connect a USB3 stick to the USB port you want to passthrough and execute the following on the Hypervisor Host:

```
$ lsusb -t
```

You will get a result similar to

```
1 /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2     |__ Port 5: Dev 18, If 0, Class=Mass Storage, Driver=usb-storage, 5000M
3 /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
4     |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
5     |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB `hostbus` value for this physical port using USB3 devices is 2 (see line 1)
The USB `hostport` value for this physical port using USB3 devices is 5 (see line 2)

12.3.3 USB hubs

If devices are connected behind a USB hub, you will see multiple nested ports behind which the device can be found.

Example 1: USB3 device behind a hub

```
1 /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2     |__ Port 1: Dev 15, If 0, Class=Hub, Driver=hub/4p, 5000M
3         |__ Port 2: Dev 16, If 0, Class=Vendor Specific Class,␣
  ↪Driver=ax88179_178a, 5000M
4 /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
5     |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
6     |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB `hostbus` value for this physical port behind a hub is 2 (see line 1)
The USB `hostport` value for this physical port behind a hub is 1.2 (see lines 2 and 3)

Example 2: USB2 device behind two hubs

```
1 /:  Bus 02.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/6p, 10000M
2 /:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd/12p, 480M
3     |__ Port 1: Dev 26, If 0, Class=Hub, Driver=hub/4p, 480M
4         |__ Port 1: Dev 29, If 0, Class=Hub, Driver=hub/4p, 480M
5             |__ Port 2: Dev 30, If 0, Class=Mass Storage, Driver=usb-
  ↪storage, 480M
6     |__ Port 6: Dev 4, If 0, Class=Wireless, Driver=btusb, 12M
7     |__ Port 6: Dev 4, If 1, Class=Wireless, Driver=btusb, 12M
```

The USB `hostbus` value for this physical port behind a hub is 1 (see line 2)
The USB `hostport` value for this physical port behind a hub is 1.1.2 (see lines 3,4 and 5)

Caution: Some USB devices contain an internal USB hub to expose multiple USB device instances. Here, the same rules apply.

12.3.4 Guest configuration

The USB `hostbus` and `hostport` value pairs need to be used in the guest configuration file `usr_guest_config.sh`.
Below you will find the required entries for the above examples.

```
# USB host passthrough (automatic passthrough for any device connected to␣
↪these ports).
#   Note: on the same physical USB port, different values for hostbus,
↪hostport pairs will show up for different USB speed!
export USB_HOST_ADAPTER1_PASSTHROUGH=""
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -
↪device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=1,hostport=4"
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -
↪device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=2,hostport=5"
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -
↪device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=2,hostport=1.2"
export USB_HOST_ADAPTER1_PASSTHROUGH="$USB_HOST_ADAPTER1_PASSTHROUGH -
↪device usb-host,bus=$USB_HOST_ADAPTER1_NAME.0,hostbus=1,hostport=1.1.2
↪"
```

Launch the guest and try to physically connect and disconnect a USB device to the configured ports. The guest should then recognize such a device automatically.

13 Graphics (Desktop) configuration

What graphical output the user will see can be configured according to the respective needs.
While during development showing the Hypervisor Host desktop (the Linux desktop) is very convenient this typically is not wanted when the system is shipped to customers.

13.1 Display both, Host and Guest Desktop

This is the default configuration after installation of the RTOSVisor.

13.2 Display Guest Desktop only

When shipping the system to customers you may want to only display the guest desktop in full screen (= kiosk) mode. If kiosk mode is enabled, the Windows (or Ubuntu) guest will run in full screen mode without the option to switch back to the Hypervisor Host desktop.

13.2.1 Guest autostart

In a production environment, the whole application should be started automatically after powering on the system.

- For how to automatically start guests, see section *Automatic guest startup*.
- The section *Standalone console mode* describes how to automatically start one of the guests showing its desktop in full screen mode (standalone console).
- If you want to use guest autostart with Hypervisor Host desktop enabled, you must enable host autologin. The chapter *Hypervisor Host autologin* describes how to *autologin* into the Hypervisor Host in desktop mode.

13.2.2 Disable Hypervisor Host desktop

When using guest autostart with the standalone console (full screen desktop view of the guest), the Hypervisor Host desktop is no longer required and should be disabled. In this case the guest is also started faster after power on.

Execute the following command to configure the Hypervisor Host into a console-only mode.

```
$ sudo systemctl set-default multi-user.target
```

After rebooting `$ sudo reboot` and configuring guest autostart in standalone mode according to *Standalone console mode*, the guest should be started automatically in kiosk mode without showing the Hypervisor Host desktop.

Hint: If you want to **reactivate the Hypervisor Host desktop**, run

```
$ sudo systemctl set-default graphical.target
$ sudo reboot
```

If you want to **temporarily start the Hypervisor Host desktop**, run

```
$ startx
```

It is even possible to completely **uninstall the Hypervisor Host GUI** to save disk space.
Please note, if the GUI is uninstalled, the interaction with the Hypervisor Host can be performed only via an SSH connection.
As an alternative, you may configure a separate Grub Menu Entry or Press Ctrl-Alt-F2 to switch to a tty2 console
The following command will completely uninstall the GUI component:

```
$ sudo apt-get purge lightdm
```

13.2.3 Monitor Mirroring (Windows guests)

If running a Windows guest in standalone console mode and more than one monitor is connected to the PC, by default the main display will be mirrored to all other displays. The display resolution will be set automatically. Depending on the connected monitors, it may be necessary to adjust the display resolution.

You will have to determine the display names and the related resolutions of the connected monitor.

```
$ export DISPLAY=:0
$ sudo xrandr
```

Below you can see a typical example where at HDMI-1 the main monitor is connected and at DP-1 the secondary one is connected.

```
$ Screen 0: minimum 320 x 200, current 1920 x 1080, maximum 16384 x 16384
$ HDMI-1 connected primary 1366x768+0+0 (normal left inverted right x axis y
 ↪  axis) 410mm x 230mm
$    1366x768      59.79*+
$    1920x1080     60.00    59.94
$    1280x1024     75.02    60.02
$    1280x720      60.00    59.94
$    1024x768      75.03    60.00
$    800x600       75.00    60.32
$    720x480       60.00    59.94
$    640x480       75.00    72.81    66.67    60.00    59.94
$    720x400       70.08
$ DP-1 connected 1920x1080+0+0 (normal left inverted right x axis y axis)
 ↪  531mm x 299mm
$    1920x1080     60.00*+
$    1680x1050     59.95
$    1280x1024     75.02    60.02
$    1280x960      60.00
$    1152x864      75.00
$    1024x768      75.03    60.00
$    832x624       74.55
$    800x600       75.00    60.32    56.25
$    640x480       75.00    59.94
$    720x400       70.08
```

If the resolution shall be adjusted, this can be set in `/hv/config/mirrormon.config`.

To set the resolution of the monitor connected at DP-1 to the same value as for HDMI-1, you may enter the following line into `mirrormon.config`:

```
DP-1:1366x768
```

Depending on the monitors connected, you may be able to set a resolution that the monitor originally did not support (in the above example, the monitor connected to DP-1 did not support 1366x768).

Caution: You should use the same resolution on all connected monitors and the resolution should by default be supported by the monitor!

Hint: If you see a black screen, it may be due to a second VM session overlapping the guest session. In this case, try switching to the correct VM session using `Alt+Tab`.

13.2.4 Monitor Switching (Linux/Debian guests)

Use case: A system which is configured to automatically start the guest in full-screen Kiosk mode after the Hypervisor Host boots. Two physical monitors are connected to the hardware with identical or different graphics resolution and **a single power source**. One of the two monitors is powered on when the VM starts. This monitor can be powered off while the second one is powered on (power switch), the guest display then will be automatically shown at the second monitor.

A template script that has to be modified is located in `/hv/guests/etc/multi-mon/switchmon-1mon.py` When the script detects that one display is connected and the first one is disconnected, it executes commands on the guest VM. Therefore, it is necessary to configure the guest VM to allow the Hypervisor Host root user to execute commands via an SSH connection without a password. Make sure to establish an SSH key exchange between the Hypervisor Host and guest. If the guest has no external access and is configured for NAT Network, the Hypervisor Host should be set up for port forwarding.

Follow the steps below to configure this script:

- Install the spice-vdagent Debian package on the Linux VM.
- Save this script to a directory within the KVM guest, e.g., `/hv/guests/guest0001`.
- Make the script executable: `chmod 755 multimon.py`.
- Modify the VM_USER variable to a user in your VM. The Hypervisor Host will execute commands in the VM using this user.

Port Forwarding

This script assumes that the VM is isolated from the external network and operates through NAT, without any external IP address. If your VM has an IP address, you need to edit and adapt this script appropriately.

Port forwarding refers to the scenario where the Hypervisor Host connects to itself through a specific port, and this connection is then forwarded to a VM. To enable port forwarding, make the following changes to `/hv/bin/kvm_guest.sh`:

Locate the following lines:

```
if [ $private_nw -eq 1 ]; then
...
USERNET=$USERNET" -netdev user,id=networkusr,smb=$HV_ROOT/guests
```

Replace them with:

```
if [ $private_nw -eq 1 ]; then
...
USERNET=$USERNET" -netdev user,id=networkusr,smb=$HV_ROOT/guests,
↪hostfwd=tcp:127.0.0
```

Hypervisor Host - Guest SSH Key Exchange

To establish a secure connection between the Hypervisor Host and VM, it is necessary to exchange cryptographic keys. Follow these steps on the Hypervisor Host:

- Switch to the root user: `sudo -i`.
- Generate host keys: `ssh-keygen -t rsa`. When prompted to provide an "Enter passphrase," press `ENTER` without entering a password.
- Copy the host key to the VM by executing the following commands:

```
ssh username@127.0.0.1 -p 8822 "mkdir -p .ssh"
cat .ssh/id_rsa.pub | ssh username@127.0.0.1 -p 8822 'cat >> .ssh/
↪authorized_keys'
```

- Replace "username" with the appropriate VM user.
- Test the connection to the VM using the following command:

```
ssh username@127.0.0.1 -p 8822
```

Verify

Perform the following steps to verify, if the script works properly:

- Start the VM.
- Execute the script: `cd /hv/guests/guest0001 && sudo ./switchmon.py`.
- Switch the power source and video cable from one monitor to another.
- The script's output should resemble the following

```
2023-05-03 16:18:04,990 Switchmon started
2023-05-03 16:18:05,012 Active monitor: DP-2
2023-05-03 16:18:07,040 Active monitor resolution: 1920x1200
2023-05-03 16:18:27,194 Active monitor: NONE
2023-05-03 16:18:31,978 Active monitor: DP-2
2023-05-03 16:18:34,007 Active monitor resolution: 1920x1200
2023-05-03 16:18:40,165 Active monitor: NONE
2023-05-03 16:18:44,937 Active monitor: HDMI-1
2023-05-03 16:18:46,962 Active monitor resolution: 1280x1024
2023-05-03 16:18:49,075 Active monitor: NONE
```

(continues on next page)

(continued from previous page)

```
2023-05-03 16:18:53,130 Active monitor: DP-2
2023-05-03 16:18:55,222 Active monitor resolution: 1920x1200
```

Automatic start

To start the script automatically in the background, make the following change to `/hv/bin/kvm_start.sh`:

Add `/hv/guests/guest0001/multimon.py&` before the `remote-viewer` line.

13.2.5 Suppress boot stage messages

Boot log

You may completely hide all logging messages while the computer boots. Adjust the files `/etc/grub.d/40_custom` and `/etc/grub.d/41_custom`. You need to edit the linux kernel boot line which includes the text "linux /boot/vmlinuz- ".

Add the following parameters to the bootline:

```
"quick splash console=ttyS0".
```

Save the file, update the grub menu and reboot.

```
$ sudo update-grub
$ sudo reboot
```

Login prompt

It is also possible to completely hide the login/password prompt that is displayed before the VM Guest window is shown in the kiosk mode. Execute the following command:

```
$ "systemctl disable getty@tty1.service"
```

and reboot.

You should see no login prompt anymore. The screen should remain black and then the VM Desktop should be displayed. If you wish to log in, press the hot key combination `Ctrl-Shift-F2..F6`.

Grub boot menu

If you want to suppress the grub boot menu, you need to edit the grub menu which is set in `/etc/default/grub`

```
$ sudo mousepad /etc/default/grub
```

Adjust these entries:

```
GRUB_TIMEOUT_STYLE=hidden
GRUB_TIMEOUT=3
```

Then you need to apply the changes:

```
$ sudo update-grub
$ sudo reboot
```

Then, during boot there will be a 3 seconds period where you have the chance to press the ESC key to show the boot menu if needed.

13.3 Guest Desktop via Pass-Through

Using graphics pass-through mode the integrated graphics hardware is assigned to the Windows guest and no longer available for the Hypervisor Host. This is described in the `Graphics Pass-Through Guide`.

14 Hypervisor Boot Customization

14.1 Splash Screen

To disable the “Splash Screen” when booting, please edit `/etc/default/grub` and update the bootloader.

```
$ cd /etc/default
$ sudo mousepad grub
```

```
remove "splash quiet" from GRUB_CMDLINE_LINUX_DEFAULT
```

and then run

```
$ sudo update-grub
```

14.2 Brand Labeling

You can find some information about how to brand label the product *here*.

15 Clone an existing Installation

You may want to clone an existing RTOSVisor installation which is prepared for production. Clone tools like bugzilla (https://clonezilla.org/) can be used for such a purpose.

Before cloning the installation, you need to remove the following files in all KVM guest folders:

- dns_conf.sh
- *_setmac.sh
- all *.log and *.bak files

16 Remote Debugging

16.1 Debugging RT-Linux Guest with Visual Studio on Windows

You may use the third party VisualGDB solution for development and debugging of RT-Linux applications using Microsoft Visual Studio. An evaluation license can be obtained here: http://visualgdb.com/download

After installing VisualGDB restart Visual Studio to get the latest VisualGDB package updates.

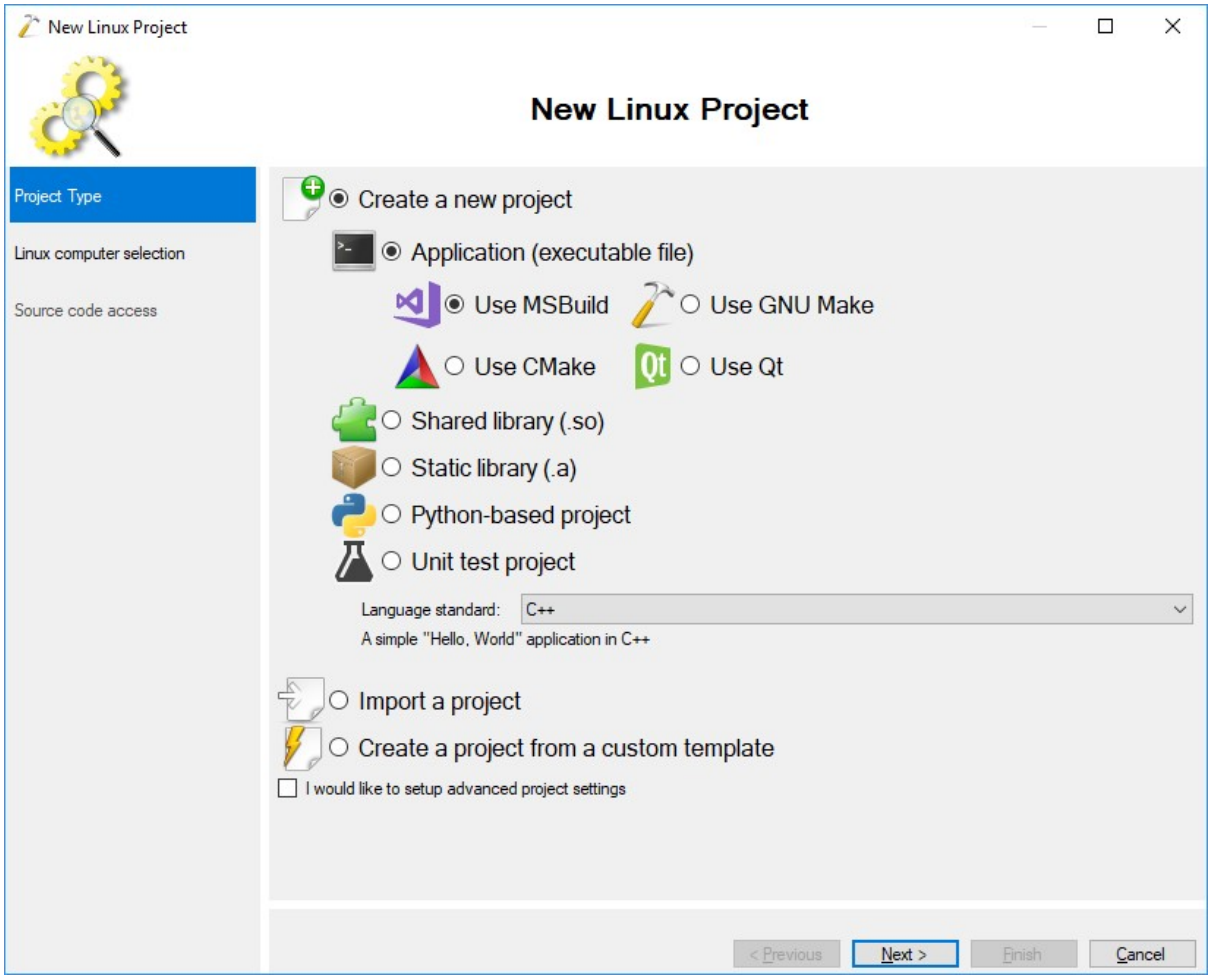
For the Kernel 5.15 both 32 and 64-bit toolchain can be downloaded here: http://software.acontis.com/LxWin/mingw64.7z

Extract the zip file into `C:\`

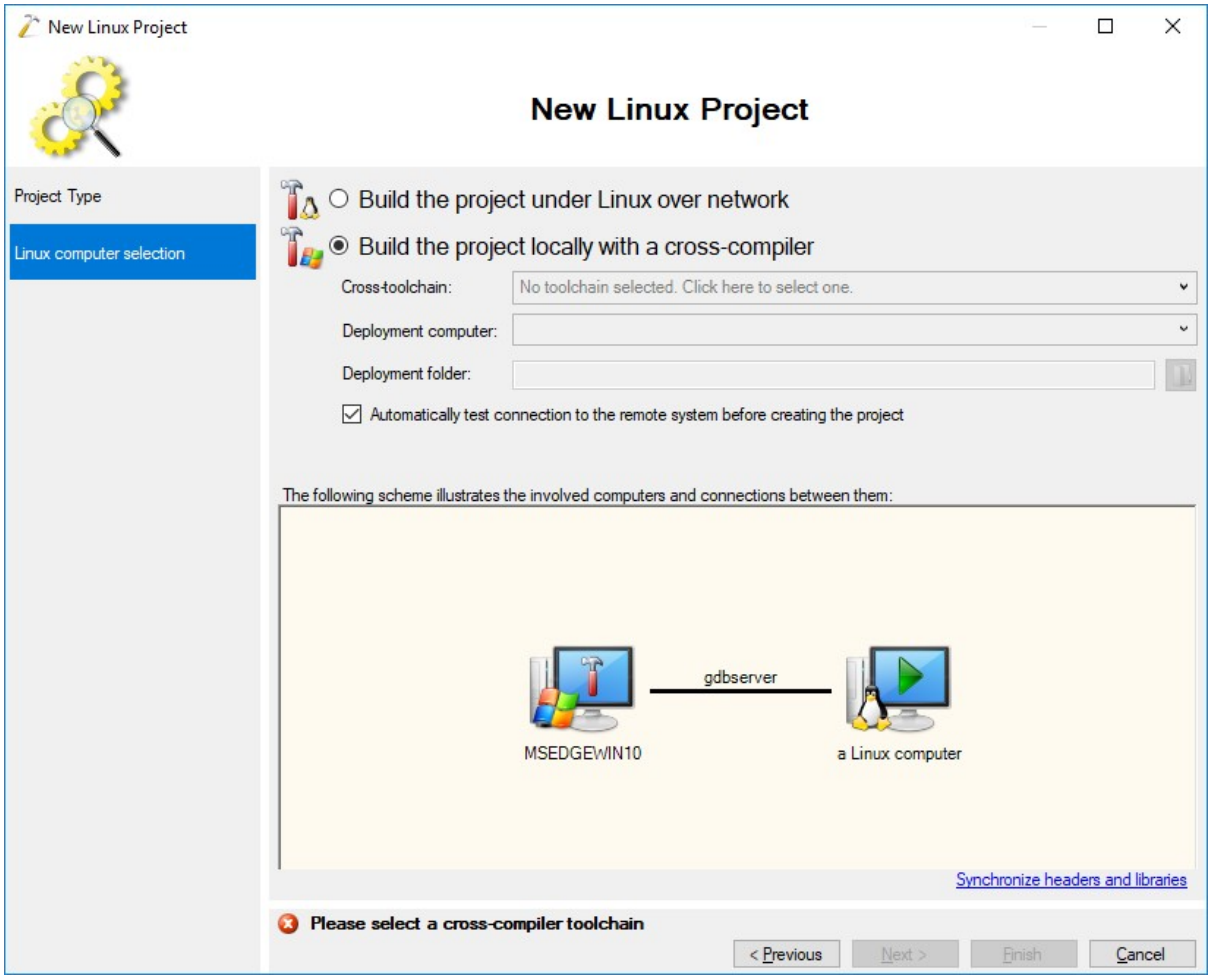
Hint: If you want to extract into another directory, ensure there are no blanks! When creating a new Visual Studio project, ensure that there are no blanks in the project path as this will produce some errors.

16.1.1 Create a new project

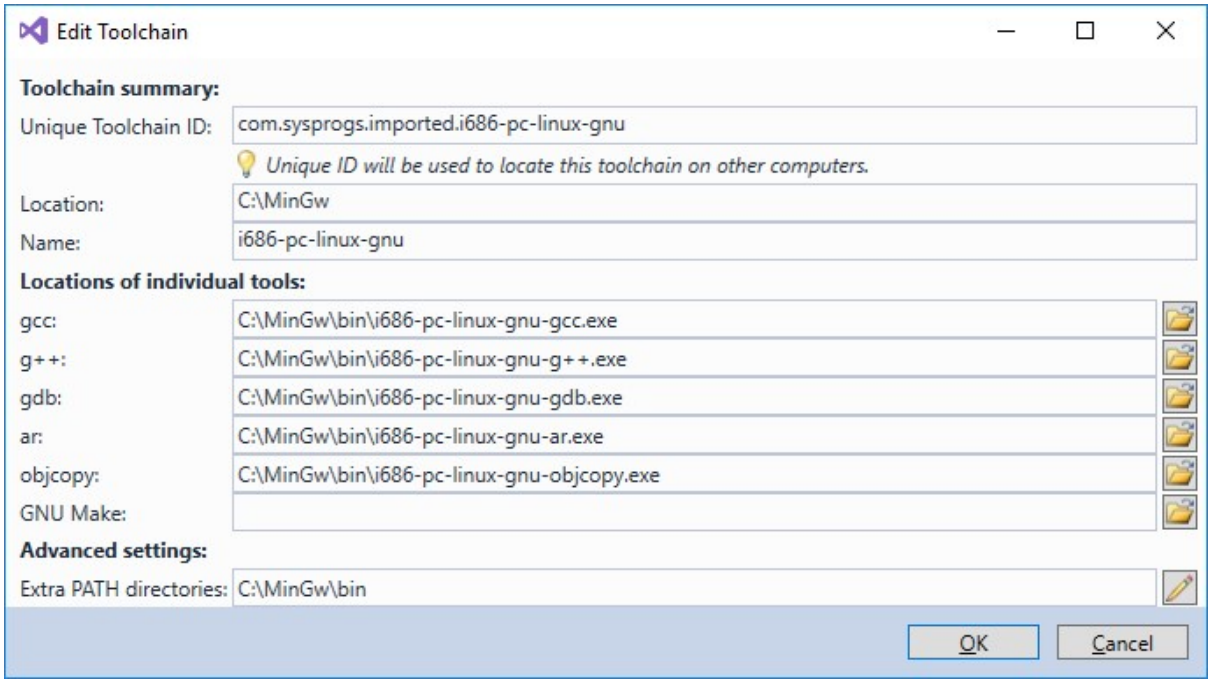
- Start the Hypervisor Host, configure to run the Linux RTOS and start RT-Linux (this is described in the Quick Start Tutorial).
- Set up network bridging and assure you can reach the RT-Linux OS from your Windows development machine. See chapter *Bridge virtual and physical network* for details. Alternatively you can also use network forwarding, see *Network Forwarding from external computer to the RTOS*
- Start Visual Studio
- Create a new VisualGDB project by using the `Linux Project Wizard`
- Set up the project as Application and use MSBuild
- Set the `Language standard` to `C++`



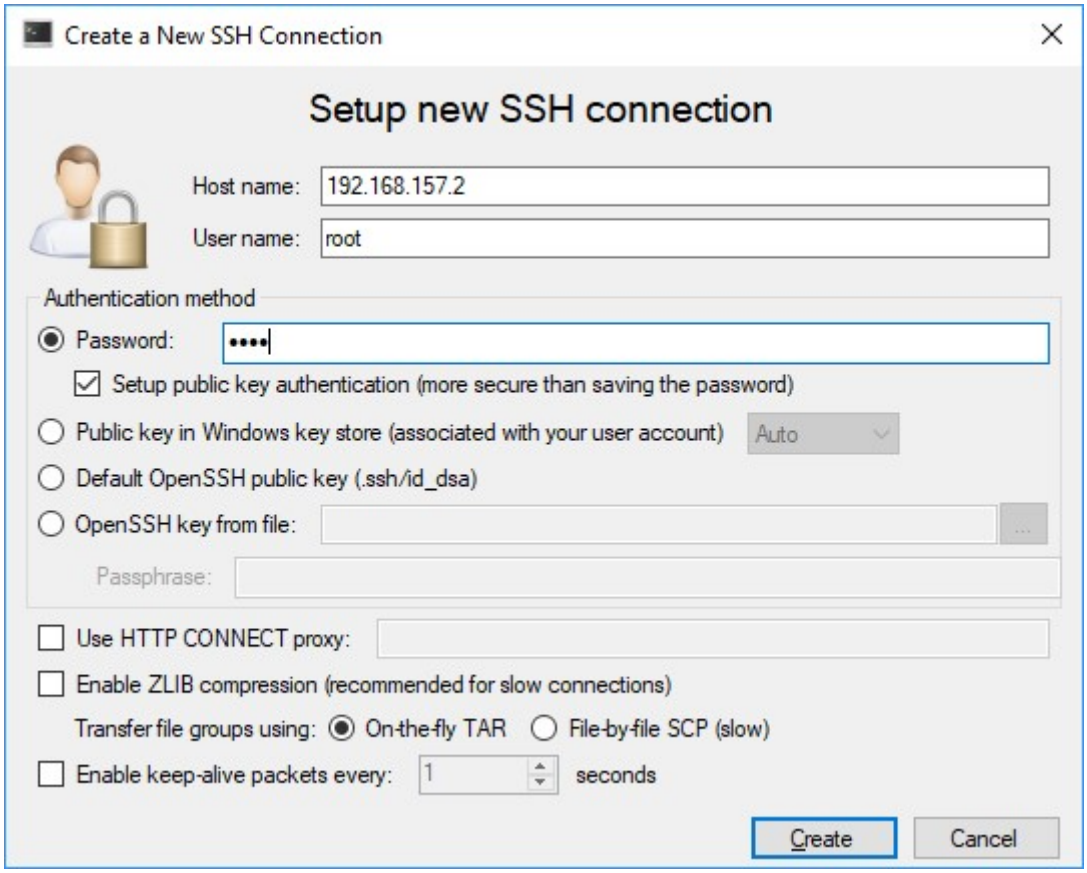
- Select `Build the project locally with a cross-compiler`



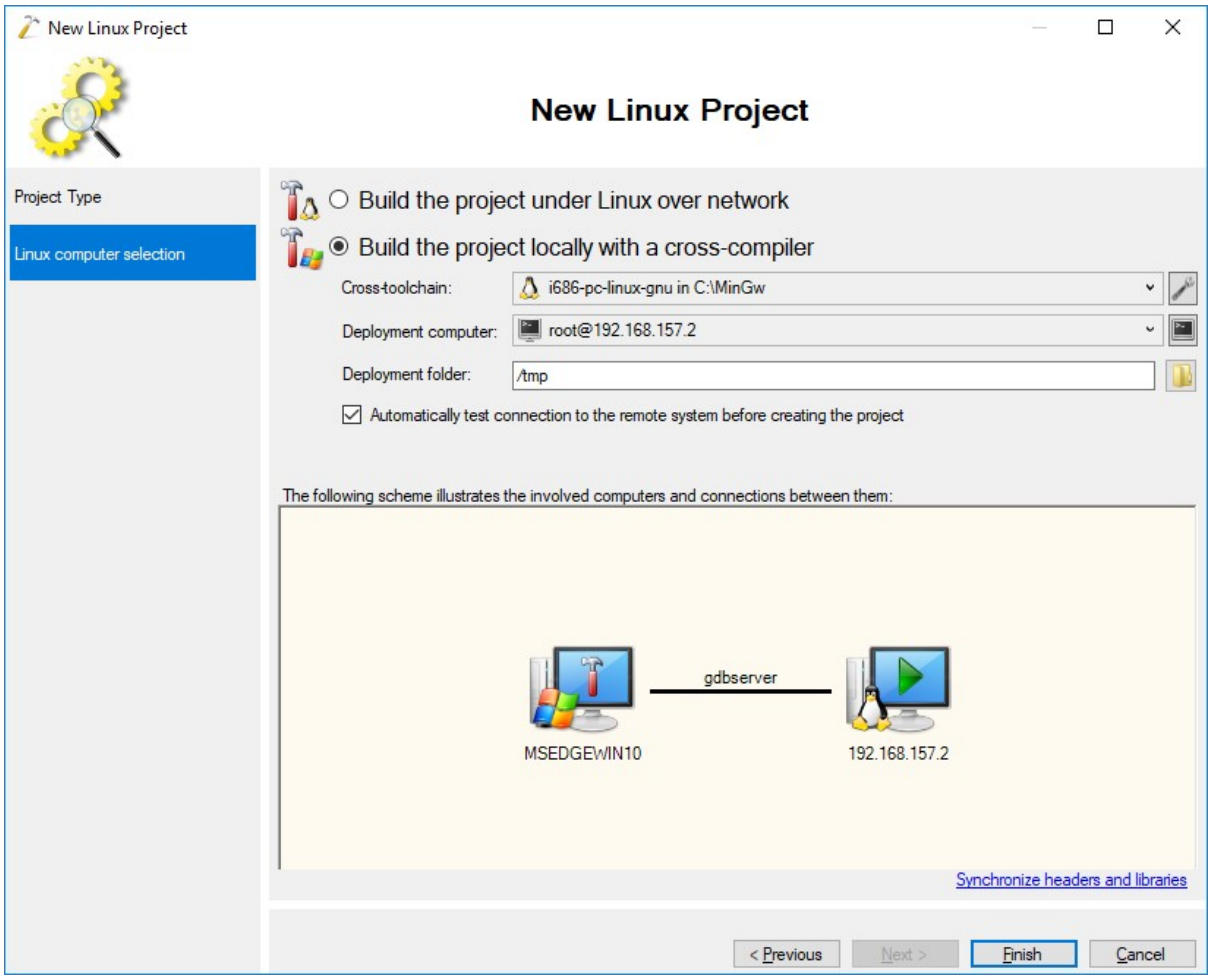
- In the Cross-toolchain field select `Locate a cross-toolchain by finding its gdb.exe` and select `C:\MinGw\bin\i686-pc-linux-gnu-gdb.exe`
- For the 64 bit (x64) toolchain select `C:\MinGw64\bin\x86_64-pc-linux-gnu-gdb.exe`
- Edit the Toolchain dialog looks like:



- In the `New Linux Project`-View, click the drop-down-field `Deployment computer` to create a new SSH connection
- Assure RT-Linux is started before you create the SSH connection! As host name use the IP address of the RT-Linux. User name and password are both `root`.



- Assure, before finishing the New Linux Project dialog looks like this:



- Then press Finish. Accept the `Mismatching environment detected` message with the OK Button.
- Now you can debug the project

16.2 Debugging RT-Linux Guest with Eclipse on Ubuntu

Setup Debug Environment

- Start the Hypervisor Host, configure to run the Linux RTOS and start RT-Linux (this is described in the Quick Start Tutorial).
- Set up network bridging and assure you can reach the RT-Linux OS from your Windows development machine. See chapter *Bridge virtual and physical network* for details. Alternatively you can also use network forwarding, see *Network Forwarding from external computer to the RTOS*
- Ensure the connection is working by establishing an SSH connection from your Ubuntu development PC.
- Install Eclipse (e.g. Eclipse Installer Package for Linux / select Eclipse IDE for C/C++ Developers)

Create a project and insert your Application Source (e.g. the RT-Linux SDK)

- Menu File - New Project - C/C++ Project
- Select Project Type (e.g. C++ Managed Build)
- Name your project and press ‘Finish’
- Extract the SDK (`/hv/guests/files/LinuxTools/rt-linux.tar`) from the Hypervisor Host into this project directory located in the eclipse workspace
- Build your examples
- Refresh the project in eclipse

Create a debug Configuration

- Run - Debug Cofigurations…
- Select C/C++ Remote Application and press the ‘New’ button

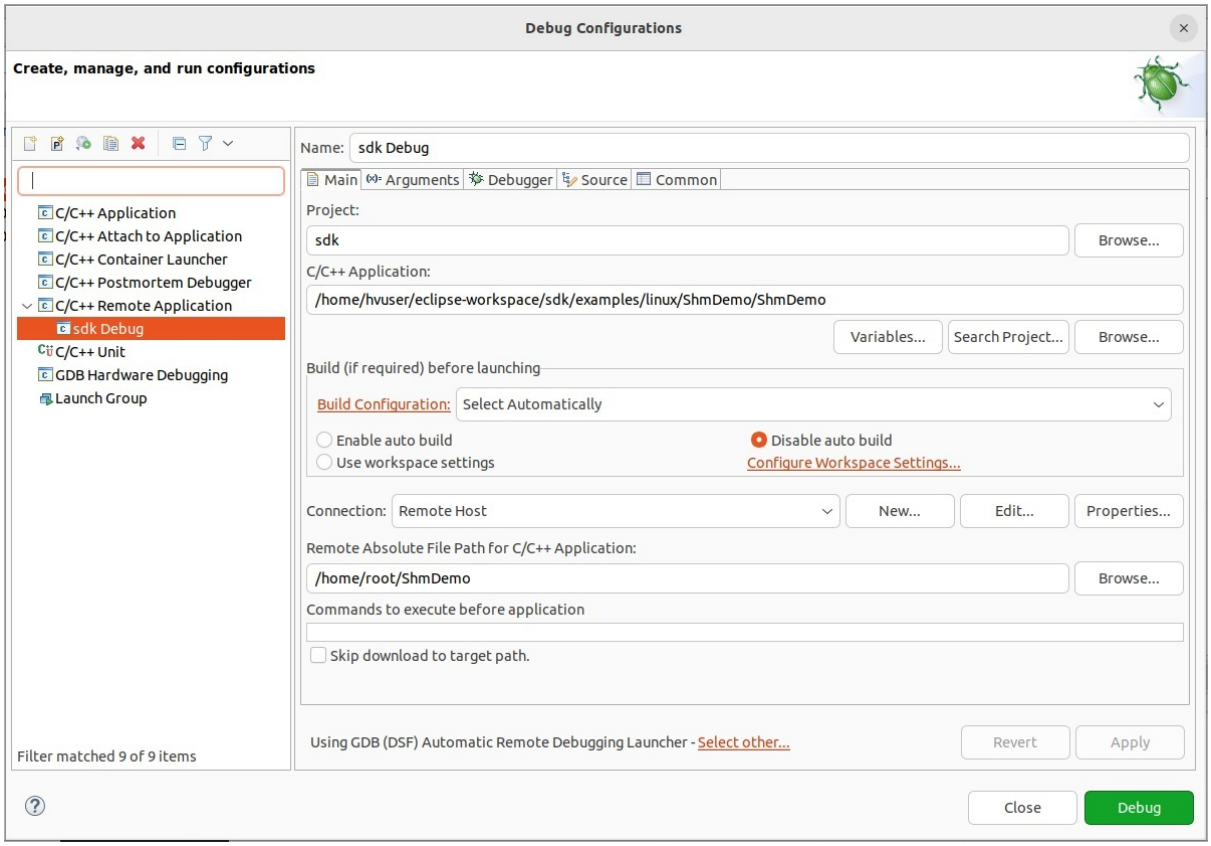
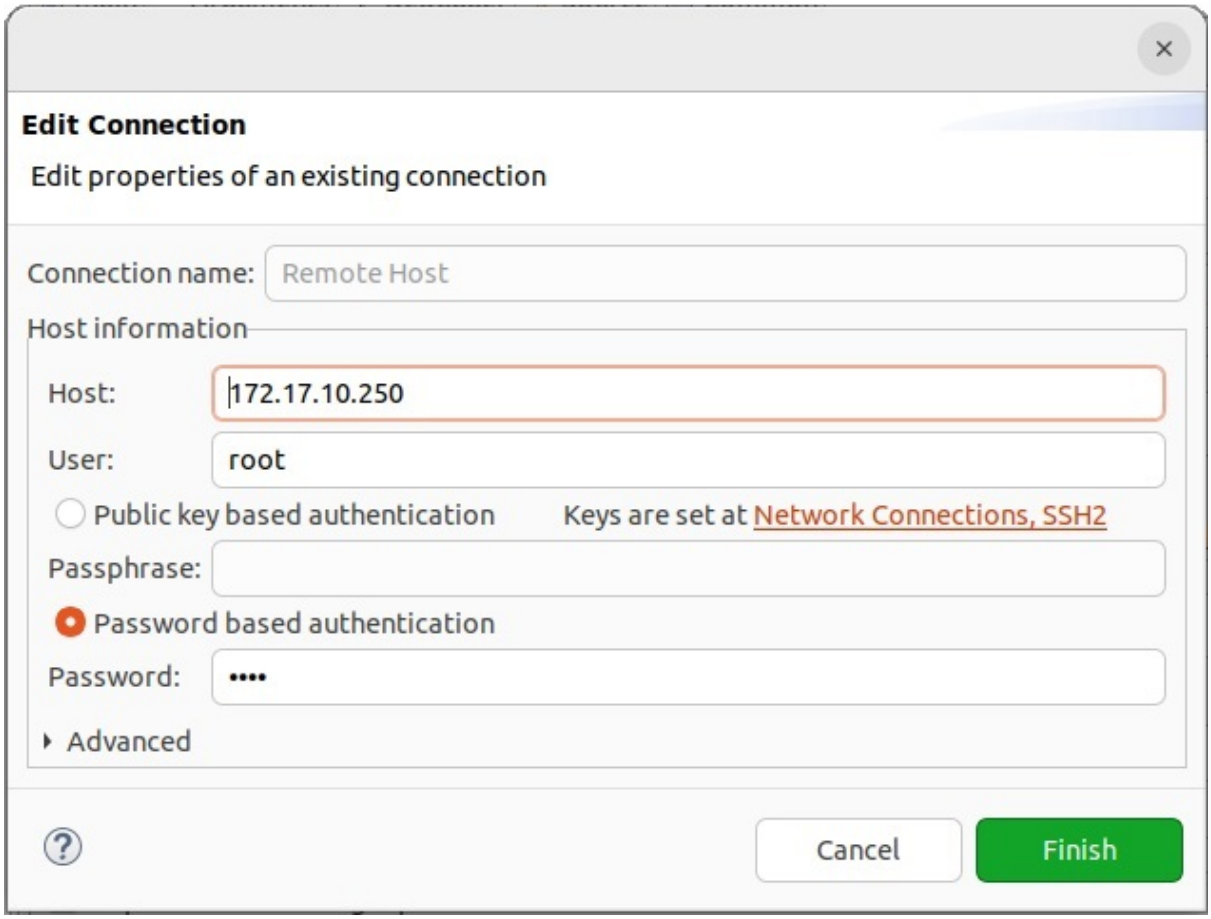


Fig. 16.1: with ssh as remote connection



- Now you are able to remote Debug your RT-Linux application

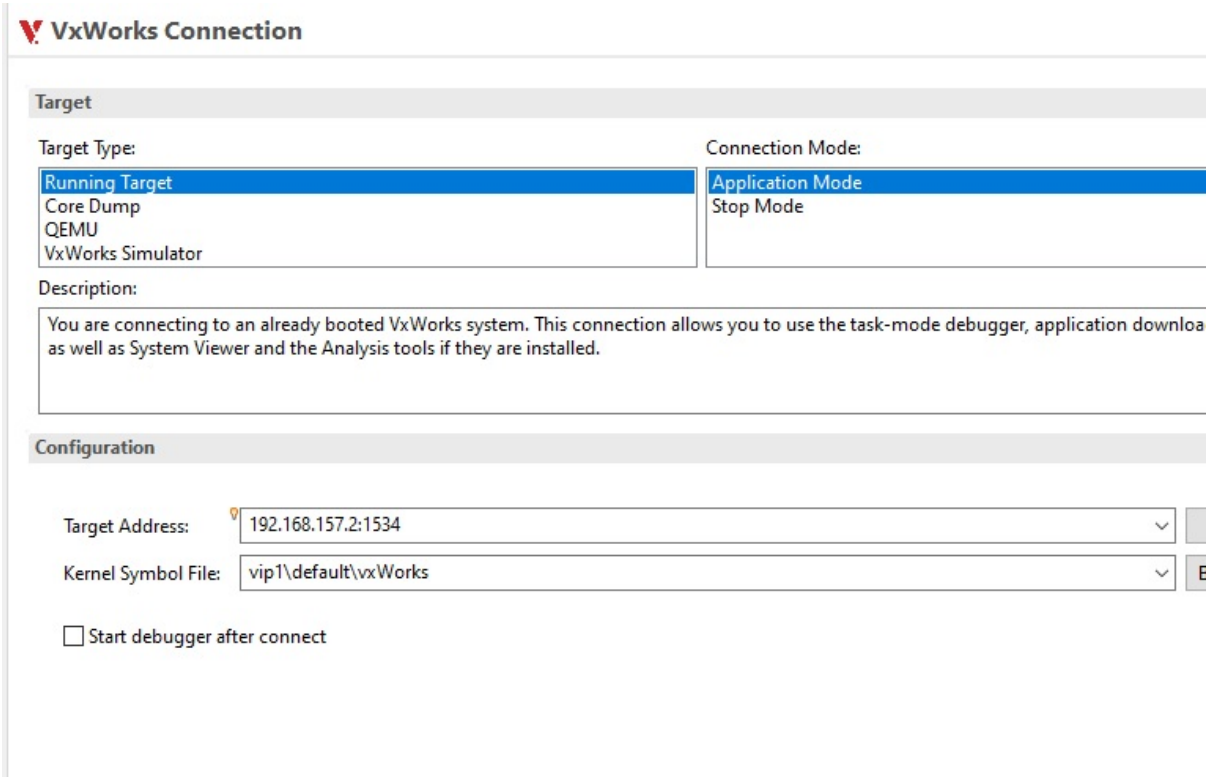
16.3 Debugging VxWorks Guest using Tornado or Workbench

- Start the Hypervisor Host, configure to run the VxWorks RTOS and start VxWorks.
- Set up network bridging and assure you can reach the VxWorks OS from your development machine. See chapter *Bridge virtual and physical network* for details. Alternatively you can also use network forwarding, see *Network Forwarding from external computer to the RTOS*
- Ensure the connection is working by establishing an SSH connection from your development PC.
- Install the VxWorks development environment.

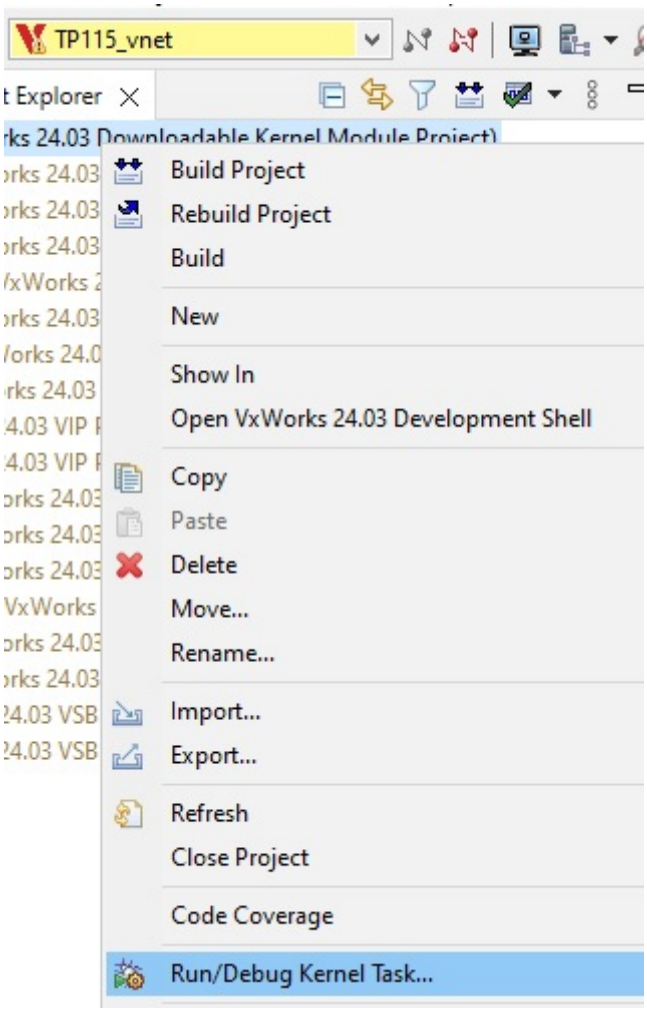
16.3.1 VxWorks7 2403

Create a downloadable kernel module and insert your Application Source (e.g. the RT-Linux SDK)

Create a VxWorks connection:



Run/Debug Kernel task:



Now you are able to remote Debug your VxWorks application

16.4 On Time RTOS-32

There are separate tutorials which describe how to set up remote debugging for On Time RTOS-32.

17 VxWorks Guest

17.1 Installation

To be able to configure the VxWorks OS and create own VxWorks binary images, the VxWorks hypervisor support files have to be copied into the VxWorks installation folder.

17.1.1 VxWorks 7 SR630

Copy the folder vxwin_common-2.0.0.1 into %WIND_HOME%\vxworks-7\pkgs_v2\os\psl\intel
Copy the folder vxwin_generic-2.0.0.1 into %WIND_HOME%\vxworks-7\pkgs_v2\os\board\intel

Create the VxWorks SR630 Source Build (VSB)

- Based on Category: All
- Based on: a VxWorks 7 board support package
- Active CPU: Pentium4
- Baseline: SR0630 2019-11-08
- BSP: vxwin_generic_2_0_0_1
- Processor mode: UP support in Libraries
- VSB Profile: Development

Create the VxWorks SR630 Image Project (VIP)

- Based on Category: All
- Based on: a source build project
- Profile: PROFILE_DEVELOPMENT

17.1.2 VxWorks7 2403

Copy the folder itl_x86_rthv_common into %WIND_HOME%\vxworks\24.03\source\os\psl\intel
Copy the folder itl_rthv_generic into %WIND_HOME%\vxworks\24.03\source\os\arch\ia\board\intel

Create the VxWorks 2403 Source Build (VSB)

- Based on Category: All
- Based on a VxWorks board support package
- BSP: itl_rthv_generic
- Active CPU: Pentium4
- VSB Profile: Minimal

Create the VxWorks 2403 Image Project (VIP)

- Based on Category: All
- Based on: a source build project
- Profile: Development

17.2 VxWorks specific configuration parameters

Default settings are stored in `guest.config` file. User can override those setting in the `usr.config` file.

17.2.1 Network configuration

All network configuration parameters are stored in `guest.config` file under the `[Rtos]` key.

Entry Name	Type	Description
Bootline	String	VxWorks boot line.
Console	String	VxWorks device name which shall be used for the console interface
VnetMACAddress	String	Virtual Network Adapter MAC address.
VnetPollPeriod	Dword	Virtual Network Adapter polling period in timer ticks.
		0 enables interrupt mode.
VnetNumCluster	Dword	Number of network clusters for the Virtual Network Adapter.
AddNetworkX	String	Additional network interface where the IP stack shall be attached to.
		X==0 (AddNetwork0): first additional interface.
		X==1 (AddNetwork1): second additional interface.
LogNetworkInterfaceX	String	Network interface for which the network packet logger shall be enabled.
		X==0 (LogNetworkInterface0): first interface for packet capturing.
		X==1 (LogNetworkInterface1): second interface for packet capturing.
LogNetworkFileX	String	File where the network packet log shall be stored.
		X==0 (LogNetworkFile0), X==1 (LogNetworkFile1), etc.
LogNetworkFileMaxSizeX	Dword	Maximum capture file size in kByte. X corresponds to the specific file.

Bootline

As shown in the following example, the user may override the default VxWorks boot line by coding his/her own under the `[rtos]` key:

```
[rtos]
"Bootline" = "vnet(0,2)pc:vxWorks h=192.168.0.1 e=192.168.0.2
u=target pw=vxworks"
```

Console

By default, the last serial channel is connected with the RTOS VM's virtual I/O channel. The VxWorks console will be redirected to the Target Console Window through the config file entry:

```
[Rtos]
"Console"="/vio/0"
```

Alternatively, the console can be redirected to any serial port – for example, the first port – using:

```
"/tyCo/0"
```

VnetMACAddress

The Virtual Network adapter address, also known as the MAC (Media Access Control) address, is generally specified in the following format:

```
"AA-BB-CC-DD-EE-02"
```

VnetPollPeriod

The Virtual Network driver by default operates in polling mode. The polling period can be adjusted using this parameter. It is set in units of the system clock timer. A value of 0 enables the interrupt mode:

```
"VnetPollPeriod" = "time units"
```

VnetNumCluster

Cluster size of the Virtual Network driver. In rare cases where extensive network traffic occurs, this parameter has to be increased:

```
"VnetNumCluster" = "size"
```

AddNetworkX

By default, the IP stack will be attached to the virtual network. If additional network adapters shall be under control of VxWorks, these adapters can be automatically attached to the IP stack using this parameter (the X has to be replaced by a contiguous number beginning with 0). The following example will attach the fei0 device (first instance of the Intel PRO/100 network adapter) to the IP stack. The IP address and subnet mask will be set accordingly:

```
[rtos]
"AddNetwork0"="fei0:192.168.100.1:255.255.255.0"
```

LogNetworkXxx

VxWin supports network packet logging which may help in solving network problems. The parameter LogNetworkInterfaceX selects the network interface where packets shall be captured. All captured packets will be stored in a file selected by the parameter LogNetworkFileX. The maximum size of the file is limited by the parameter LogNetworkFileMaxSizeX (kByte). The format of the file is compatible with PCAP:

```
[rtos]
"LogNetworkInterfaceX" = "interface name"
"LogNetworkFileX" = "filename"
"LogNetworkFileMaxSizeX" = "max size in kBytes"
```

17.3 Filesystem access via FTP

The filesystem of the hypervisor can be accessed via FTP. You need to adjust the VxWorks bootline so it matches the username and password of the hypervisor.

You may use the shell commands to verify if FTP access works:

```
$ cd "/host.pc"
$ ls
```

17.4 Filesystem access via NFS

An alternative to the FTP file access is using NFS. In a first step, the NFS server must be installed in the Hypervisor host, see *NFS access* The VxWorks image then must include NFS client support.

You will have to determine the UID and GID of the folder that is exported. In this example, we assume the `/hv/guests` folder is exported.

```
$ cd /hv
$ ls -ldn guests
```

The resulting output will show the UID and GID of the folder:

```
drwxr-xr-x 12 1000 1000 4096 Mai 23 19:53 .
```

- The first column *drwxr-xr-x* represents the permissions of the folder.
- The second column *12* is the number of hard links.
- The third column *1000* is the UID of the folder owner.
- The fourth column *1000* is the GID of the group owner.

In VxWorks, network authentication can be set then using *nfsAuthUnixSet*. The following example will use the host name *pc* which is determined in the bootline and the UID/GID determined before:

```
$ nfsAuthUnixSet("pc",1000,1000,0,0)
```

Using the *nfsMount* call a NFS share can be mounted:

```
$ nfsMount("pc","/hv/guests","/guests")
```

Alternatively, you can configure the VIP (VxWorks Image Project) to mount all available NFS shares automatically using the appropriate UID/GID values.

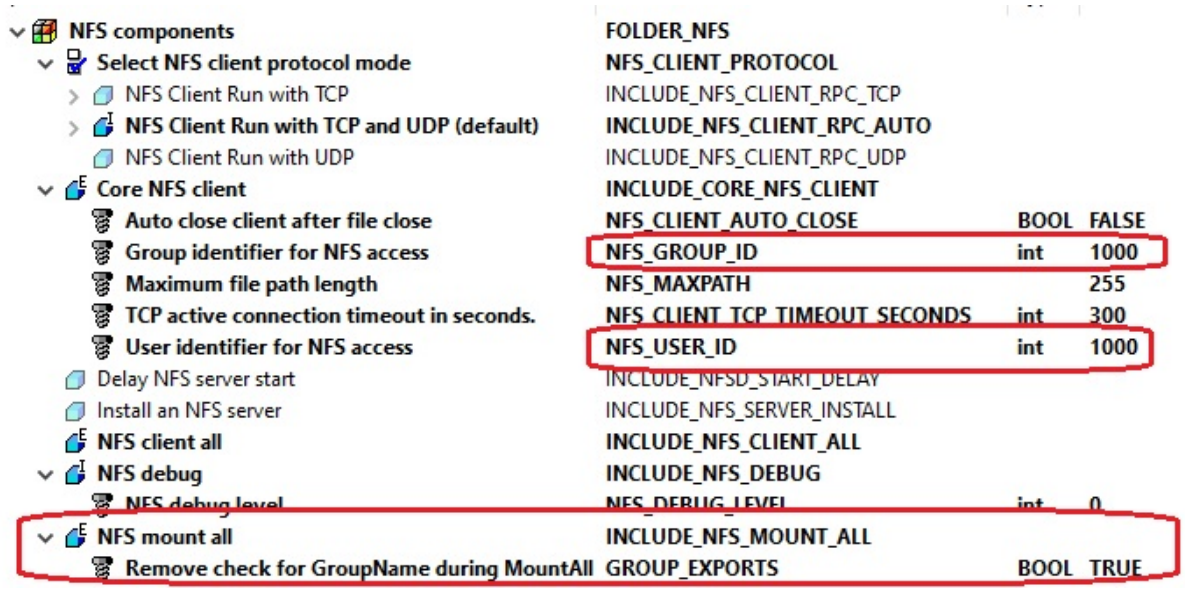


Fig. 17.1: NFS configuration.

17.5 Autostart

You can place a startup script (e.g. `startup.sh`) in the guests sub folder `files`, e.g. into `/hv/guests/guest0001/files`. Then you need to add the `s=/host.pc/hv/guests/guest0001/files/startup.sh` parameter into the bootline (stored in `guest.config` in the guest folder).

This startup script then may load additional object modules and run an application. An example script may look as follows:

```
$ cd "/host.pc/hv/guests/guest0001/files"
$ ld<objmodule1.out
$ ld<objmodule2.out
$ sp myApp
```

17.6 RTOS Library

The RTOS library provides higher-level communication services for synchronizing Windows with VxWorks or to exchange data between the operating systems. The RTOS library is based on VMF-functions, which provide the basic communication functionality. A description of the RTOS Library can be found in the document RtosVM User Manual (see the acontis online Development Center and look for the VxWin manuals).

18 Filesystem access and file sharing

This chapter provides information about how guests can access the filesystem which is under control of the Hypervisor host.

18.1 Default SMB share

The Hypervisor Host by default exposes the `/hv/guests` folder to KVM guests via a SMB share (Windows file share). The share can be accessed from within the guest using the IP address `10.0.2.4` and the share name `qemu`. For example on Windows 10 it can be accessed through `\\10.0.2.4\qemu`. The exposed folder is set in the `/hv/bin/kvmguest_start.sh` script in parameter `smb` of the `USERNET` configuration.

```
if [ $private_nw -eq 1 ]; then
    # user network
    USERNET="-device virtio-net,netdev=networkusr,mac=$ethmacVM2"
    USERNET=$USERNET" -netdev user,id=networkusr,smb=$HV_ROOT/guests"
    echo "private network MAC = "$ethmacVM2
else
    echo "no virtual network"
fi
```

18.2 SMB (Windows) file share

Instead of using the (limited) default `SMB` share, you may use `SAMBA` to get a more flexible solution.

You need to configure the `SMB` server properly. It is recommended to use the same username for the network share as you are using for the Hypervisor Host. To determine the user, you may run:

```
$ whoami
```

In this document we assume, the username is `hvuser`.

If you want use a different user for the SMB share, this user must also be configured for the Hypervisor Host. For example, to add a new user `smbuser`, run the following command:

```
$ sudo adduser smbuser
```

To create a file share which is accessible from a remote (Windows) computer, go to the `SAMBA` configuration file:

```
$ sudo mousepad /etc/samba/smb.conf
```

In the below example, we will create a share with the name `guests` (`[guests]`) which will share the folder `/hv/guests`. We will add the Hypervisor host username (`hvuser`) as well as the additional user `smbuser`.

Add the following section to the end of the `smb.conf` file and save:

```
[guests]
  comment = guests share
  path = /hv/guests
```

(continues on next page)

(continued from previous page)

```
  browseable = yes
  valid users = hvuser, smbuser
  guest ok = yes
  read only = no
```

If you encounter issues with the file share, you may also adjust the following section in the `smb.conf` file:

```
[global]
map to guest = never
```

Then you need to provide network share access for the user `hvuser` and `smbuser`:

```
$ sudo smbpasswd -a hvuser
$ sudo smbpasswd -a smbuser
```

Restart the `SAMBA` service:

```
$ sudo systemctl restart smbd.service nmbd.service
```

Verify, if the users have been correctly added:

```
$ sudo pdbedit -L
```

To check, if the share is active, try to access the `SAMBA` share from a Windows computer using the explorer. You may have to use the IP address of the hypervisor.

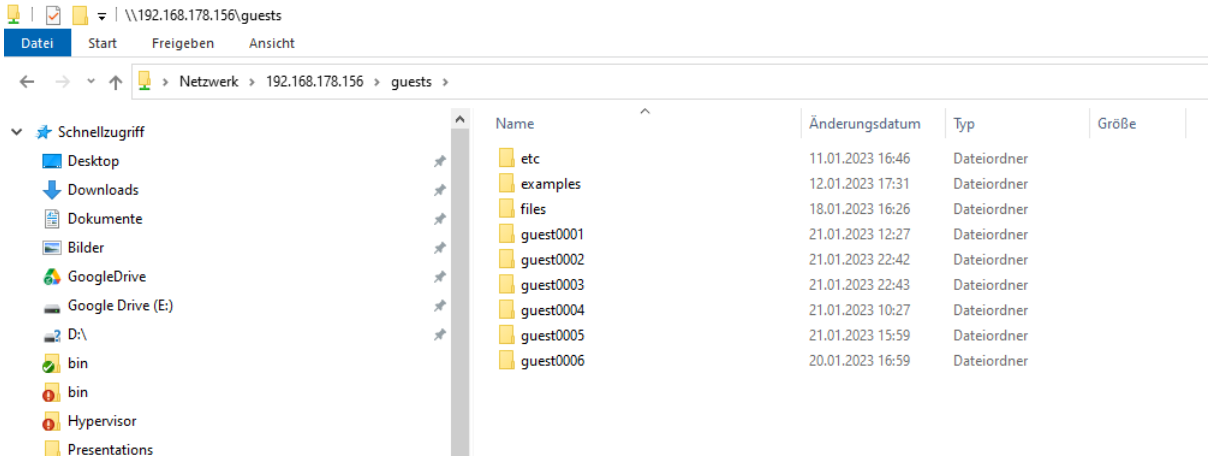


Fig. 18.1: Access to share from Windows file explorer.

18.3 FTP access

The vsftpd FTP server is running on the Hypervisor Host. You can login using the username and password of the Hypervisor. FTP clients can be used to exchange files between guests or external computers and the Hypervisor. Write access is disabled by default. You can enable it by editing the `/etc/vsftpd.conf` configuration file (uncomment the `write_enable=YES` line).

18.4 SSH access

A Secure Shell server (SSH) is running on the Hypervisor Host. SSH or SFTP clients can be used to exchange files between guests or external computers and the Hypervisor.

18.5 NFS access

For some guests (e.g. VxWorks) it may be necessary to use the NFS (Network Filesystem) protocol.

You will have to determine the folder you want to share with guests. In this example, we assume the `/hv/guests` folder shall be shared.

Edit the Exports File

```
$ sudo mousepad /etc/exports
```

At the end of this file, add the directory you want to share, followed by the network range that should have access and the permissions. For example:

```
/hv/guests *(rw,sync,no_root_squash,no_subtree_check)
```

Apply the Export Configuration

```
$ sudo exportfs -a
```

Start and Enable NFS Service

```
$ sudo systemctl start nfs-kernel-server
$ sudo systemctl enable nfs-kernel-server
```

Verify NFS Server

```
$ sudo exportfs -v
```

18.6 Sharing removable devices

Using the `hv_diskshare` tool it is possible to share removable devices like USB sticks or Compact Flash cards with multiple guests.

The `hv_diskshare` tool is an integral part of the hypervisor solution, providing robust file-sharing capabilities for removable devices. The tool is capable to mount the file systems on the hypervisor and expose the file system via NFS (network file system) and/or SMB (server message block). The exposed file systems then can be accessed from within one or multiple guests. This chapter outlines the features and usage of the `hv_diskshare` tool to manage NFS and SMB shares for removable devices.

18.6.1 NFS shares (Network File System)

Network File System (NFS) is a distributed file system protocol originally developed by Sun Microsystems (Sun). Fore more details about using NFS shares on Windows guests, see *Using the NFS share on Windows guests*. Fore more details about using NFS shares on Linux guests, see *Using the NFS share on Ubuntu/Debian guests*.

18.6.2 SMB shares (Windows filesharing protocol)

Server Message Block (SMB) is a communication protocol used to share files, printers, serial ports, and miscellaneous communications between nodes on a network. It is typically used in Windows. Fore more details about using SMB shares on Windows guests, see *Using the SMB share on Windows guests* Fore more details about using SMB shares on Linux guests, see *Using the SMB share on Ubuntu/Debian guests*

18.6.3 Mounting and sharing removable devices

- USB filesystems are mounted at `/mnt/diskshare/usb`, for example `/mnt/diskshare/usb/sda1`
- SATA filesystems are mounted at `/mnt/diskshare/sata`, for example `/mnt/diskshare/sata/sr0`
- The share name under which the filesystem will be exposed to guests will be the physical device name (e.g. `sda1` or `sr0`).

Inside a Windows or Linux guest, if you want to determine which filesystems are exposed, you can use the following commands:

- Show NFS shares in Linux/Windows clients: `showmount -e <IP-Address>`
- Show SMB shares in Linux clients: `smbclient -L IP-Address -N`
- Show SMB shares in Windows clients: `net view \\IP-Address`

18.6.4 Installing and Removing NFS and SMB Servers

The tool supports the installation and removal of both NFS and SMB servers, ensuring that the necessary services are available for sharing devices. By default, both servers should already be installed.

Install NFS Server:

```
hv_diskshare --nfs
```

This command installs the NFS server if it is not already installed.

> **Caution:** By default, this function does not work as modifying the installation is disabled.

Remove NFS Server:

```
hv_diskshare --rmnfs
```

This command removes the NFS server if it is installed.

Install SMB Server:

```
hv_diskshare --smb
```

This command installs the SMB server if it is not already installed.

> **Caution:** By default, this function does not work as modifying the installation is disabled.

Remove SMB Server:

```
hv_diskshare --rmsmb
```

This command removes the SMB server if it is installed.

18.6.5 Listing Removable Devices

To list all removable devices connected to the hypervisor, use the following command:

```
hv_diskshare --devs
```

This provides a detailed list of all removable devices, including their labels, device nodes, and other relevant information.

18.6.6 Managing Shares

The *hv_diskshare* tool allows for dynamic sharing and unsharing of devices via NFS and SMB.

Share All Available Removable Disks:

```
hv_diskshare --shareall
```

This command mounts and shares all available removable disks.

Share a Specific Device:

```
hv_diskshare --share devname
```

Replace *devname* with the specific device name (e.g., `sdb1`) to mount and share only that device.

Mount a Specific Device:

```
hv_diskshare --mount devname
```

This mounts a specific device without sharing it.

Unmount a Specific Device:

```
hv_diskshare --unmount devname
```

This unmounts a specific device.

Stop Sharing All Devices:

```
hv_diskshare --stopall
```

This command stops sharing all currently shared devices and unmounts them.

Stop Sharing a Specific Device:

```
hv_diskshare --stop devname
```

This stops sharing and unmounts a specific device.

18.6.7 Dynamic Device Monitoring and Sharing

The script can dynamically monitor devices for changes, such as additions or removals, and handle sharing and unsharing as needed.

Enable Dynamic Monitoring:

```
hv_diskshare --dyn
```

This command starts the observer to monitor block devices and trigger device events dynamically. Log messages show details about added or removed devices, mounted filesystems and exposed network shares.

If you want to run the observer as background task, run:

```
hv_diskshare --dynback
```

Once the background observer mounted and shared a device, it will be kept until the device is removed or `hv_diskshare --stopall` or `hv_diskshare --stop my_devicename` is called. The log file which includes all observed events is located in `$HV_BIN/pylib/diskshare.log`. Print the log file content:

```
hv_diskshare --printlog
```

Remove the log file:

```
hv_diskshare --dellog
```

To stop running the background observer, run:

```
hv_diskshare --stopdyn
```

Filesystem access from Windows or Ubuntu guests

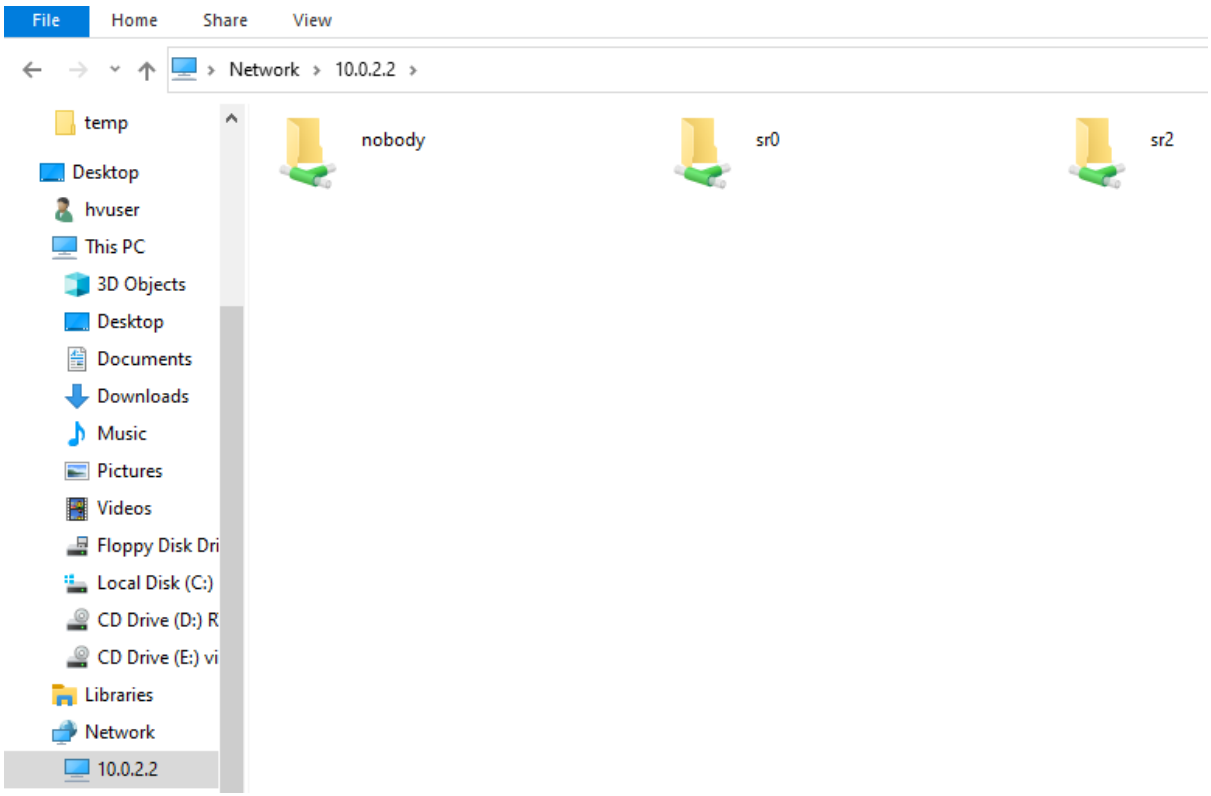
Each time when a device is added or removed, a python script `$HV_BIN/pylib/diskshare_hook.py` is called which can be used to start a user specific action. For example, an application can be started which sends a message to a specific guest to notify that guest about a new device being connected or disconnected.

Alternatively, you may also share the `/mnt/diskshare` folder to guests via NFS or SMB (see *SMB (Windows) file share* or *NFS access*). Then all removable devices will be automatically visible inside the guest whenever they are connected.

Windows guests

SMB

The SMB network shares will be accessible via the `10.0.2.2` internal network:



You can also run the command `net view \\\\10.0.2.2` to see which SMB shares are available.

Caution: In case of errors, please check Windows security settings: *Using the default SMB share on Windows guests*.

NFS

If you want to use NFS in a Windows guest, you need to enable the **Client for NFS** Windows feature.

You need to create the following two files (replace `username` by the appropriate Windows username):

- `C:\Windows\System32\drivers\etc\passwd`

```
username:x:1000:1000:username:c:\users\username
```

- `C:\Windows\System32\drivers\etc\group`

```
username:x:1000:1000
```

Then you can run the following command to see which NFS shares are available:

```
$ showmount -e 10.0.2.2
```

To mount or unmount a NFS share, run the following command (mounting drive letter `Y:` and assuming the `showmount` command returned `/mnt/diskshare/usb/sda1`):

```
mount \\10.0.2.2\mnt\diskshare\usb\sda1 Y:
net use Y: /delete
```

Linux guests

NFS

In case you have uninstalled the NFS client, re-install the NFS client again:

```
$ sudo apt update
$ sudo apt install nfs-common
```

Then you can run the following command to see which NFS shares are available.

```
$ showmount -e 10.0.2.2
```

To mount a NFS share, run the following commands (assuming the `showmount` command returned `/mnt/diskshare/usb/sda1`):

```
$ sudo mkdir -p /mnt/usb/sda1
$ sudo mount -v -o vers=4,nolock,proto=tcp 10.0.2.2:/mnt/diskshare/usb/sda1
↪  /mnt/usb/sda1
```

SMB

In case you have uninstalled, re-install the SMB client again:

```
$ sudo apt update
$ sudo apt install samba-client cifs-utils
```

Then you can run the following command to see which SMB shares are available:

```
$ smbclient -L 10.0.2.2 -N
```

To mount a SMB share, run the following commands (assuming the `smbclient` command returned `sda1` as share name), replace `myuser` with the actual username:

```
$ sudo nano /etc/fstab
$ add this line at the end of the file: //10.0.2.2/sda1
↪  /home/myuser/mnt/usb  cifs  user,noauto,guest  0  0
$ sudo systemctl daemon-reload
$ mkdir -p /home/myuser/mnt/usb
$ mount -t cifs //10.0.2.2/sda1 /home/myuser/mnt/usb -o user=myuser
```

19 Miscellaneous

19.1 Installing additional packages

By default, it is not possible to install additional software packages (debian packages) through an Internet connection.

> **Caution:** Please note, downloading and then using such additional packages will lead to losing support and warranty for the product!

Only for debugging or development purposes you may enable to download additional packages. For that purpose, you need to replace the file `/etc/apt/source.list` by `/etc/apt/source.list.online`.

19.2 Error (Support Information) Report

In case of errors or technical questions, you may generate an error report which will collect various system informations. This report can be sent to the acontis technical support for further investigation.

Run the following command to create the error report:

```
$ hv_gen_error_report
```

You may also generate such report using the System Manager tool (Help - Download Support Info).

19.3 Installing Hypervisor updates

From time to time you may get updates for the Hypervisor, for example a debian package that has to be installed or a single file that needs to be replaced. Unless such updates are not automatically installed you need to follow the specific instructions for the update.

In a production environment, it is recommended to completely disconnect the Hypervisor host from the network. This is described in *Network airgapping*.

If you want to update the Hypervisor over the network while the Hypervisor host is airgapped from the network, you may run a dedicated update service guest. Such guest can be temporarily executed for update purposes and be connected to the network even though the Hypervisor host is not able to access the network.

The following steps show a possible update scenario which keeps the Hypervisor host completely disconnected from the external network:

- Stop all running applications and guests.
- Start the update service guest. This guest must have access to the Hypervisor's private (internal) network and mount the default SMB share: *Default SMB share*. This is an internal Hypervisor network share which is not exposed to the external network. Additional information about filesystem access and how to mount SMB shares can be found in *Filesystem access and file sharing*

- For security reasons, you should disable this private network before connecting the service guest to the external network.
- Download the update via the service guest and store it there.
- For security reasons, disconnect the service guest from the external network.
- Enable the private network to access the Hypervisor filesystem.
- Store the update files in the Hypervisor filesystem.
- Initiate the update installation in the ypervisor host (e.g. by creating a file on the default SMB share that triggers an update sequence in the Hypervisor host).
- Stop the update service guest.
- Run the update in the Hypervisor host, if necessary, reboot the system.

19.4 Virtual Memory and SWAP space

When your system runs out of physical RAM, the Linux kernel can move (or "swap") inactive pages from memory onto disk. This mechanism helps prevent out-of-memory (OOM) errors by freeing up RAM for active tasks. However, using swap on disk is slower than using physical RAM, so it is best used as an overflow or safety net rather than a primary source of memory.

To manage swap, RTOSVisor includes a script called **hv_swap_mgr**. This script allows you to create, resize, remove, and query a **swapfile** as well as inspect existing **swap partitions** and **memory/RAM usage**. Below is an overview of its features:

- **Create or resize swapfile** Use `-sfs <size_in_MB>` to set up a persistent swapfile. - Enforces a minimum swap size of **1GB** and a maximum of **total RAM**. - Automatically updates `/etc/fstab` to ensure the swapfile is active after reboot.
- **Remove swapfile** Use `-sfrm` to deactivate and remove the swapfile, and remove any matching entry from `/etc/fstab`.
- **Query swap partition** Use `-spq` to check if there is a swap partition. If found, it displays the total size and current usage as a percentage.
- **Query swapfile** Use `-sfq` to see how much of the **swapfile** (not partition) is currently used. Internally, it subtracts swap partition usage from the total system swap to isolate the file usage.
- **Query memory usage** Use `-mq` to show an overview of system RAM usage, including total, used, free, buffer/cache, and available values (in both MB and percent).
- **Combined queries** - `-sq` performs both partition and file swap queries. - `-q` performs partition and file swap queries **plus** a memory usage query.

By default, **hv_swap_mgr** uses `/swapfile` as the swapfile location, but you can override this by setting the environment variable `SWAPFILE` before running the script:

```
export SWAPFILE=/my/custom/swapfile
hv_swap_mgr -sfs 2048
```

Refer to the built-in help (`-h`) for detailed usage and examples.

19.5 Updating Firmware

RTOSVisor is based on thoroughly tested hardware configurations and uses stable official hardware driver versions in its distribution.

But we are not able to test every new version of every driver for every instance of new hardware.

Therefore, on the one hand, our distribution includes only official tested drivers and does not allow installing untested new versions, but on the other hand, we want to give the user the opportunity to test new hardware before the official release of its drivers.

This mechanism is made possible through the separation of hardware software into two parts: drivers and firmware. Drivers are built into the real-time kernel itself and are updated by us. And firmware, which are .bin files lying in the /lib/firmware directory, can be updated.

Please note that the use of software not tested by us is at your own risk and acontis does not guarantee the operation of new firmware. Downloading and then using such additional packages will lead to losing support and warranty for the product!

To update or install new firmware: Step 1. Please read the section “Installing additional packages” and follow the steps to activate the ability to install additional software described in this section.

Step 2. Execute script:

```
/hv/bin/update-firmware.sh
```

19.6 KVM Guests with SecureBoot

Caution: This is currently not supported!

19.6.1 UEFI and Legacy BIOS

19.6.2 UEFI

What is SecureBoot

Secure Boot is an interface between UEFI and Operating System. When SecureBoot is activated, it prevents the loading of unsigned boot loaders or drivers

Read More: https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

Preparing keys for SecureBoot

The key thing in SecureBoot is a platform key (Platform). It establishes a relationship between a platform owner and a platform firmware. PK is a self-generated certificate owned by OEM.

Another important key is a KEK key. This key is obtained from an OS manufacturer (for ex. Microsoft) and is used to establish trust relationship between the firmware and OS.

Generating the platform key:

```
openssl req -newkey rsa:2048 -nodes -keyout PKpriv.key -x509 -days 365 -
↪out PK.crt
Generating a 2048 bit RSA private key
....+++
.+++
writing new private key to 'PKpriv.key'
-----
You are about to be asked to enter information that will be incorporated␣
↪into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Bayern
Locality Name (eg, city) []:Munic
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Beer Inc
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: BayernBeer
Email Address []:
```

OVMF supports keys in DER format only. So we need to convert it:

```
$ openssl x509 -in PK.crt -outform der -out PK.der
```

Download Key Exchange Key (KEK): `MicCorKEKCA2011_2011-06-24.crt` https://go.microsoft.com/fwlink/p/?linkid=321185

Download Signature Database (allows Windows to boot): `MicWinProPCA2011_2011-10-19.crt` https://go.microsoft.com/fwlink/?LinkId=321192

Download Microsoft signer for third party UEFI binaries via DevCenter: `MicCorUEFCA2011_2011-06-27.crt` https://go.microsoft.com/fwlink/p/?LinkID=321194

Building OVMF with SecureBoot support

By default, OVMF is built without SecureBoot support.

So it is recommended to fetch this project from its repository and build OVMF yourselves.

Install required packages:

```
$ sudo apt-get install build-essential git uuid-dev iasl nasm -y
$ sudo apt-get install iasl -y
$
$ git clone git://github.com/tianocore/edk2.git
$ cd edk2
```

Prepare build tools:

```
$ git submodule update -init
$ make -C BaseTools
$ .edksetup.sh
$ make -C ./BaseTools
$ export EDK_TOOLS_PATH=/home/rte/edk2/BaseTools
$ .edksetup.sh BaseTools
```

Edit `Conf/target.txt`:

```
ACTIVE_PLATFORM = OvmfPkg/OvmfPkgX64.dsc
TARGET_ARCH = X64
TOOL_CHAIN_TAG = GCC5
```

Build OVMF with SecureBoot support:

```
$ OvmfPkg/build.sh      -a IA32 -a X64      -D SMM_REQUIRE -D
↪  SECURE_BOOT_ENABLE      -D FD_SIZE_2MB -D EXCLUDE_SHELL_FROM_FD
```

Binaries can be found in the `Build` directory.

Embedding SecureBoot keys to OVMF

Create a `OVMF-SecureBoot` directory and copy `Build/OvmfX64/DEBUG_GCC5/FV/OVMF_CODE.fd` and `Build/OvmfX64/DEBUG_GCC5/FV/OVMF_VARS.fd` to this directory.

Create an `hda` subdirectory and copy all generated and downloaded keys to this subdirectory.

Run qemu:

```
$ cd OVMF-SecureBoot
$ qemu-system-x86_64 -L .      -drive
↪  if=pflash,format=raw,readonly,file=OVMF_CODE.fd      -drive
↪  if=pflash,format=raw,file=OVMF_VARS.fd      -hda fat:hda      -net none
```

After booting you get to a UEFI shell. Type `exit`.

1. Go to Device Manager / Secure Boot Configuration / Secure Boot Mode and change from **Standard Mode** to **Custom Mode**.
2. PK Options / Enroll PK / Enroll PK Using File and choose `PK.der`
3. KEK Options / Enroll KEK / Enroll KEK Using File and choose `MicCorKEKCA2011_2011-06-24.crt`
4. DB Options / Enroll Signature / Enroll Signature Using File and choose `MicWinProPCA2011_2011-10-19.crt`
5. Repeat last step and choose `MicCorUEFCA2011_2011-06-27.crt`

The Secure Boot Mode should be **Enabled** now.

Exit from BIOS, shutdown the machine.

19.7 RTOSVisor Host SecureBoot support

Caution: This is currently not supported!

It is also possible to install RTOSVisor on a host with activated Secure Boot.

To achieve this, we suggest the following steps:

1. Deactivate Secure Boot
2. Install RTOSVisor.iso
3. Install the acontis certificate in UEFI
4. Activate the acontis bootloader
5. Activate Secure Boot
6. Use the acontis certificate for mass production of hardware

19.7.1 Deactivating Secure Boot

It is worth noting that each computer manufacturer installs different UEFI firmware on their computers, which may result in differences in the Secure Boot activation and deactivation procedure. In this article, we will highlight general considerations using ASUS (R) UEFI BIOS Utility (TM) as an example. In other UEFI firmware, this procedure is done in a similar way.

When the computer boots, you may wish to press a hot key to enter your BIOS. In our case, this is a “Delete” key.

In BIOS, please navigate to the Boot menu and then to the “Secure Boot” menu.

Here you can see the status of Secure Boot: Disabled or Enabled. If it is enabled, we kindly request that you disable it. To do so, please navigate to the “Key Management” sub-menu and select the “Clean Secure Boot Keys” command.

Then save your changes and reboot. Secure Boot should be disabled in this case.

19.7.2 Installing RTOSVisor.iso

Install it as usual.

19.7.3 Installing the acontis certificate in UEFI

The UEFI Firmware must trust the acontis digital signature in order for the acontis boot loader to take control and run the operating system in protected mode.

To do this, add a digital certificate from Acontis to the UEFI database located in your computer’s NVRAM.

Boot into the RTOSVisor system. Select the boot menu “Configure RTOSVisor” or “Hypervisor”.

The next step is to run the sb-addkey.sh script located in the /boot/efi/EFI/acontis directory

```
$ cd /boot/efi/EFI/acontis
$ sudo ./sb-addkey.sh
```

After running the script, you will be prompted to set a password. This is needed to allow UEFI to add the acontis certificate to NVRAM after reboot. Set the password, then confirm it. After that, reboot the computer.

If everything was done correctly, after rebooting you will see a big blue screen with the title “Perform MOK management”.

Select the “Enroll MOK” menu. Then “Continue”. Then select “Yes” to the question “Enroll the key(s)?”. Then enter your password, you’ve set before. Then select “Reboot” on the next screen.

19.7.4 Activating the acontis bootloader

Before activating the acontis bootloader, please make sure the acontis certificate has been successfully added to your UEFI.

```
$ mokutil --list-enrolled
```

The acontis certificate must be present in the list of certificates issued by this utility.

Usually it is located in the [key 2] section.

If everything is correct, please now activate the acontis bootloader.

```
$ cd /boot/efi/EFI/acontis
$ sudo ./sb-activate.sh
```

Once activated, the bootloader acontis becomes the default bootloader on your system. UEFI then checks the bootloader’s digital signature, ensuring it is known to UEFI, and starts the bootloader. The acontis boot loader then checks the digital signature of the second-stage boot loader, acontis grub2, which is known to the acontis boot loader.

If everything is normal, acontis grub2 bootloader, digitally signed by acontis, searches for the operating system kernel from acontis. It finds it and also checks the digital signature. The acontis kernel is digitally signed by acontis, so the bootloader trusts it and starts it.

Then the OS kernel, after initializing all subsystems, starts loading numerous drivers. Each driver in the acontis distribution has also been digitally signed by acontis, so they will also be loaded. This ensures that all components are loaded securely at every stage and that no component has been tampered with or modified by a third party.

please reboot your computer after activation of the acontis bootloader.

19.7.5 Activating Secure Boot

The steps to activate Secure Boot are similar to the previous section on deactivating Secure Boot.

When the computer boots, press a hot key to enter your BIOS. In our case, this is a “Delete” key. In your case it can be F1, F5, F10 or F11 or something else.

In BIOS, please navigate to the Boot menu and then to the “Secure Boot” menu.

Here you can see the status of Secure Boot: Disabled. Then please navigate to the “Key Management” sub-menu and select the “Install Default Secure Boot Keys” command.

Every hardware vendor adds many keys to the system when installing UEFI Firmware. These are either trusted or untrusted certificates. There are several levels of keys used for this purpose: PK, KEK, DB, DBX. The “Install Default Secure Boot Keys” command loads all these default keys into NVRAM.

Then go back to “Boot->Secure Boot” menu again and make sure that the “Secure Boot state” is now “Enabled”.

Save your changes and restart your computer again.

Then select one of the boot menus, either “Configure RTOSVisor” or “Hypervisor” menu and make sure that the operating system boots successfully.

After booting, make sure that Secure Boot is indeed activated. To do this, run the command:

```
$ sudo bootctl
```

find the line “Secure Boot” and make sure it’s “Enabled”.

19.7.6 Use the acontis certificate for mass production of hardware

If you are a hardware manufacturer and you decide which UEFI Firmware configuration to install, or you have a contact with your hardware manufacturer who can do so, then simply ask for or add the acontis certificate to the DB database that is pre-installed with the UEFI Firmware. This will ensure that the user of your hardware does not need to go through all the steps described above. RTOSVisor can simply be installed and started using it immediately.

Location of acontis certificate in DER format: /boot/efi/EFI/acontis/cert.der

19.8 Performance Optimizations

19.8.1 KVM Guest preallocation

KVM uses the QCOW2 disk image format to store a guest system’s virtual disk as a file on the host, representing the guest’s filesystem and data. By default, KVM uses sparse allocation for QCOW2 disk images. This means that disk space is allocated dynamically as data is written, starting with a small file that grows in size only when required. While this approach is space-efficient initially, it can lead to file fragmentation over time, which may negatively impact performance, especially under heavy I/O workloads.

To address this, preallocation can be used for QCOW2 images. Preallocation reserves disk space upfront, reducing fragmentation and improving I/O performance. With full preallocation the entire disk size is allocated at creation, ensuring the best I/O performance at the cost of higher initial space usage and longer creation time.

After the initial startup of your KVM guest, the QCOW2 image is created in sparse mode by default. To convert the QCOW2 image to a fully preallocated format, stop you KVM guest and follow these steps:

```
$ # change to the path with your qcow2 file
$ cd /hv/VMs/windows
$
$ # create preallocated copy of your qcow2 file
$ qemu-img convert -O qcow2 -o preallocation=full windows.qcow2
 ↪  windows.pre.qcow2
$
$ # check if the preallocated qcow2 was created without error
$ qemu-img info windows.pre.qcow2
$
$ # replace the existing qcow2
$ mv windows.pre.qcow2 windows.qcow2
```

20 RTOS-32 Legacy information

Caution: This documentation is not valid for RTOSVisor V8.1 and later!

The below documentation is an excerpt from RTOSVisor V8.0. Some of the files can be found in `/hv/guests/etc/rtos-32`.

20.1 On Time RTOS-32 container

- **Directories:**

  `/hv`
  this root directory contains all RTOSVisor files and executables.

  `/hv/rtos-32`
  RTOS-32 configuration files and start/stop scripts as well as OS binaries.

  `/hv/rtos-32/rtfiles`
  directory for your `.dlm` files.

- **Most important bash scripts:**

  `/hv/rtos-32/realtimedemo.sh`
  starts RTOS-32 VM and the acontis tool measuring context switch and interrupt latencies.

  `/hv/rtos-32/realtimedemo-debug.sh`
  starts debug monitor that awaits requests from a remote debugger on a Windows PC with installed EcWin, Visual Studio, and RTE Plugin.

  `/hv/rtos-32/ecmasterdemo.sh`
  starts RTOSVisor RTOS-32 VM and EtherCAT MasterStack demo.

  `hv_guest_stop`
  stops VM and RTOSVisor.

  `/hv/rtos-32/dbgcon.sh`
  opens interactive RTOS-32 VM console, so you can interact with your app here.

- **Specific command used for debugging:**

  `hv_brvnetset`
  creates a virtual network bridge on Hypervisor Host to forward debugger *TCP/IP/UDP* packets from `LAN1` to RTOS-32 VM. It is required to start this script if you need to perform remote debugging of an RTOS-32 app from another machine.

  Hint: See chapter “Bridge virtual and physical network” in the Hypervisor Manual for details how to configure the bridge.

  ---

hv_vnetclr
deletes bridge, after the RTOS-32 VM has been stopped.

Start an RTOS-32 container using the `rtos-32.sh` script.

20.1.1 How to run a sample preconfigured RTOS-32 App (Realtimedemo)

RTOSVisor has a special tool (`RealtimeDemo`), which can accurately measure the real-time capabilities of a machine and the hypervisor. This tool can be also used as a first sample realtime app, to play with the RTOSVisor.

To start it, execute `/hv/rtos-32/realtimedemo.sh` script. This script loads/starts RTOSVisor VM, RTOS-32 OS Image and starts `RealtimeDemo.dlm`. Execute `hv_guest_stop` script to stop everything.

The most important parameters here are: *Interrupt Delay* and *Task Delay* (in microseconds). These parameters show the realtime capabilities of the Hypervisor Host and RTOSVisor.

You can press `Ctrl` + `C` to exit from the console into Linux Terminal. Please execute `dbgcon.sh` script to return to the RTOS-32 console again.

It is also easily possible to run the realtime demo in the background, completely detached from the console. So, use `dbgcon.sh` script in this case, to open interactive RTOS-32 terminal from a new console.

20.1.2 How to run `EcMasterDemo` that uses a network card and the EtherCAT stack

It is assumed that your Hypervisor Host has two ethernet ports `LAN1` and `LAN2`. The first one can be used for a *TCP/IP* traffic and/or to establish a debug connection with a DevPC.

It is also assumed that your second `LAN` port was already assigned to RTOSVisor, as it is described in this `Hypervisor.pdf` in the **PCI/PCIe device assignment** section.

Execute the following steps:

1. Connect `LAN2` port to a EtherCAT slave.
2. `cd /hv/rtos32`
3. `sudo ./ecmasterdemo.sh`

All required files like `EcMasterDemo.dlm`, `EcMaster.dlm`, `emllRTL8169.dlm` are already in the `rtfiles/ecmaster` subdirectory.

20.1.3 How to remotely debug an RTOS-32 app from a Windows PC with Visual Studio

It is assumed that you also have a second Windows PC (DevPC), where you installed Visual Studio + EcWin + RTE Plugin. DevPC is used to develop your RTOS-32 Application and perform a remote debugging of this app on the Hypervisor Host.

acontis has developed a special Visual Studio plugin that provides the possibility to create an RTOS-32 project, compile, debug it (local or remote) and configure a Visual Studio environment for you.

Requirements for a Windows PC:

* Visual Studio 2015 or newer should be installed

* acontis Rtos32Win/EcWin should be installed

Topics like creating a new RTOS-32 Project are out of scope of this document. Please refer to the official acontis RTOS-32 documentation to find details.

Before we start, it is important to understand how RTOS-32 apps work in the RTOS-32 VM. Every app consists of a Loader part (`/hv/rtos-32/Loader.bin`) and a dynamically loaded module in a `.dlm` format (`/hv/rtos-32/rtfiles/yourapp.dlm`). `Loader.bin` is provided with full source code and comes as a separate Visual Studio project. It is automatically generated by the RTE Visual Studio Plugin using the method described above.

However, an application that is supposed to be debugged, should use a separate bootloader called **RTOS-32 Monitor** (`/hv/rtos-32/debug/Monvmf.bin`).

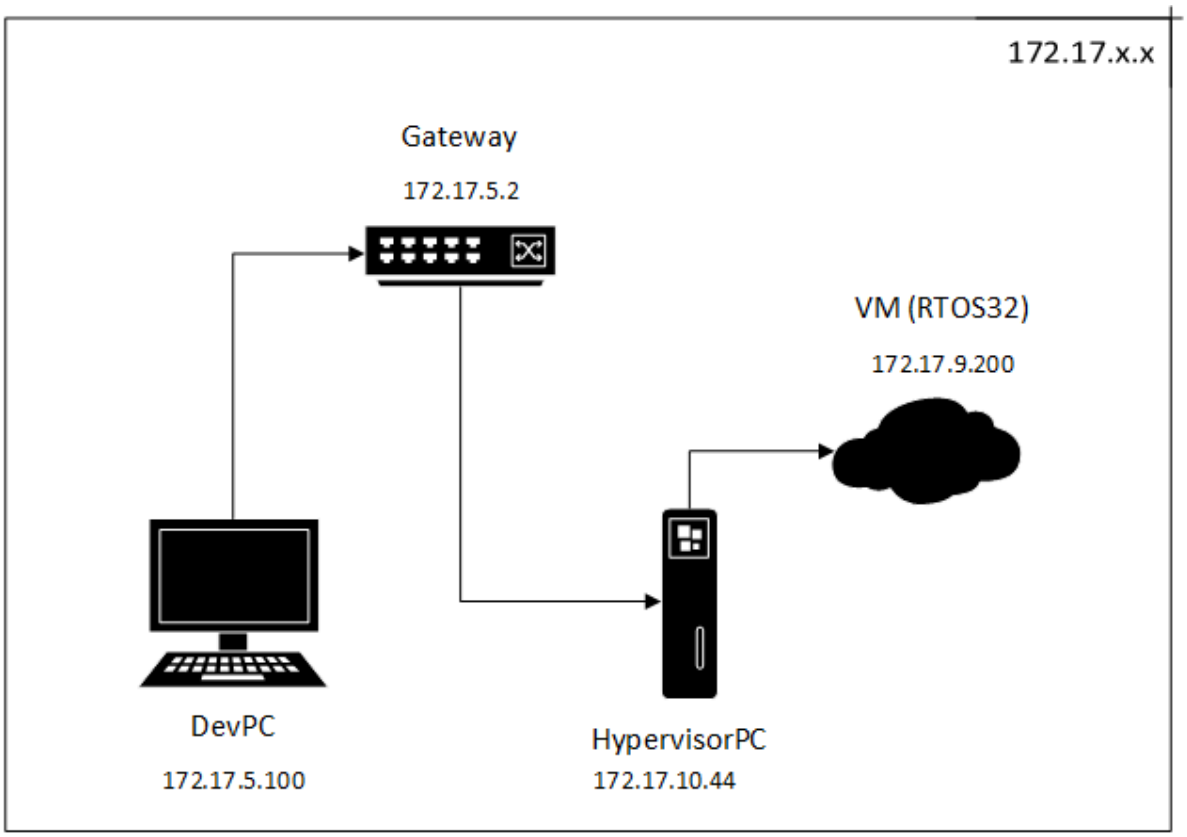
The monitor starts in RTOS-32 VM and listens to a *TCP/IP* traffic from the remote Visual Studio debugger.

The second important thing: all Monitor settings (like an *IP* address to listen to) are always embedded into the `Monvmf.bin` file. So, copy `Monvmf.bin` to the Hypervisor Host from DevPC every time when you change remote *IP* address in the System Manager (the part of EcWin/Rtos32Win).

Network Configuration:

The Hypervisor Host should be configured to perform the remote debugging; scripts and configuration files should contain specific *IP* addresses/masks/subnets. The requirement is that DevPC, Hypervisor Host and RtosVM are in the same subnet, in our example `172.17.x.x`.

Our sample network configuration:



All these *IP* addresses should be set here: `/hv/config/brvnetconfig.sh`

In our example, the gateway is a simple router with running DHCP server, that assigns static `172.17.5.\*` addresses to development PCs in the office and dynamic `172.17.10.\*` addresses to all other computers connected to the router.

VM with address `172.17.9.200` is in a separate subnet, not physically connected to the router. But Hypervisor Host accepts `MASK 172.17.255.255` on its virtual network bridge (read below) and can forward all traffic to and from a VM from the `172.17.x.x` network.

In simple words, the DevPC communicates always with VM. In the VM a special debug stub is started (called **RTOS-32 Debug Monitor**) that listens to the `172.17.9.200` *IP* address.

Open a project of a sample demo app (we describe here how to debug `RealtimeDemo`):

1. DevPC: open System Manager and right click `RTOS #1\Application` and choose `Create New Application Project (Debug Only)`
2. Choose `RealtimeDemo` and click *OK* button. Visual Studio projects are generated now and copied to `RtFiles` and other subdirectories of your System Manager workspace.

Prepare a debugger bootloader (RTOS-32 Monitor):

1. DevPC: Select *Remote Debugging* and click *Settings* button near to a grayed *IP* text field.
2. Enter `172.17.9.200` in the **IP** text field and press *OK* button. Please note, now the new RTOS-32 Monitor binary (`Monvmf.bin`) is generated and the target *IP* addresss is embedded into the binary.

Compile the `.dlm` project:

1. DevPC: Click *Open Project with Visual Studio* button.
2. In the Visual Studio two projects are available in the solution explorer: `Loader` and `RealtimeDemo`.
3. Right click every project and click *Build*

Copy binaries to the Hypervisor Host:

1. Copy `RealtimeDemo.dlm` from the `RtFiles` directory in your workspace on DevPC to `/hv/rtos-32/rtfiles/debug/` directory.
2. Copy `projects/monvmf/Monvmf.bin` to `/hv/rtos-32/debug/` directory
3. Make sure `/hv/rtos-32/realtimedemo-debug.config` file has correct settings. It should have a valid file server directory (`hv/rtos-32/rtfiles/debug/`) and it should of course include `.config` file for a PCI Card for `LAN2` port, that is assigned to the RTOS VM.

Start RTOS-32 VM and the RTOS-32 Monitor (Hypervisor Host):

1. Execute `/hv/rtos-32/realtimedemo-debug.sh`. Please note, the `.dlm` module was already added to the `realtimedemo-debug.config` in the `Rtos\Loader` section.
2. You should see an RTOS-32 Monitor output. It listens to the IP address `172.17.9.200` to receive the remote debugger traffic.

Please check all your network settings here: `/hv/config/brvnetconfig.sh`

Please note, vnet0 virtual network adapter is created on the Hypervisor Host, when the VM is started. Please consider this adapter as a virtual PCI network card inside the RTOS-32 VM, that is visible in Hypervisor Host as `vnet0`.

By default the `192.168.178.1` address is assigned to this adapter on a Linux side.

Configure a virtual bridge network adapter

Configure a virtual bridge network adapter to forward all incoming debugger traffic from `LAN1` to the virtual network adapter `vnet` in the RTOS VM.

Please execute `/hv/hvctl/brvnetset.sh` script. The virtbr network adapter is created on the linux side.

Start debugging (DevPC):

1. In Visual Studio open the file `Loader.cpp`.
2. Locate the `main()` function
3. Set a breakpoint.
4. Press `F5` key to start debugging.

How to repeat/stop debugging:

1. There is no need to restart the RTOS-32 Monitor if you want to stop the debugging.

Simply click Stop in the Visual Studio and then press `F5` again.

1. It is not needed to copy the RTOS-32 Monitor binary (`Monvmf.bin`) every time you make changes in the loader or a `.dlm` project. Only when the *IP* address is changed.
2. **If you found a bug in your `.dlm` module and need to upload a new verison into the Hypervisor Host, please do the following:**

    1. compile the project
    2. copy changed `.dlm` binary to the `/hv/rtos-32/rtfiles/debug/` directory.

3. **If you want to completely stop your VM, execute the following sequence:**

    1. Execute `/hv/hvctl/brvnetclr.sh` to remove the bridge
    2. Execute `hv_guest_stop`