ctys-**QEMU/KVM**(7)
Installation and **Configuration**

November 29, 2010

Contents

List of Tables

List of Figures

.

1 QEMU/KVM - Basics for Operations

The ctys-QEMU plugin supports the emulation of various CPUs by QEMU as well as it's accelerator modules e.g. KVM and KQEMU(under development). The KVM accelerator of the Linux kernel is handled as a specific accelerator thus supported by the QEMU plugin.

The ctys-QEMU plugin of the UnifiedSessionsManager supports a subset of the QEMU command line options mapped to native options, whereas remaining options are just bypassed. Therefore a meta-layer for an abstract interface is defined, which is implemented by a wrapper script. The wrapper script is written in bash syntax and sourced into the runtime process, but could be used for native command line calls as well.

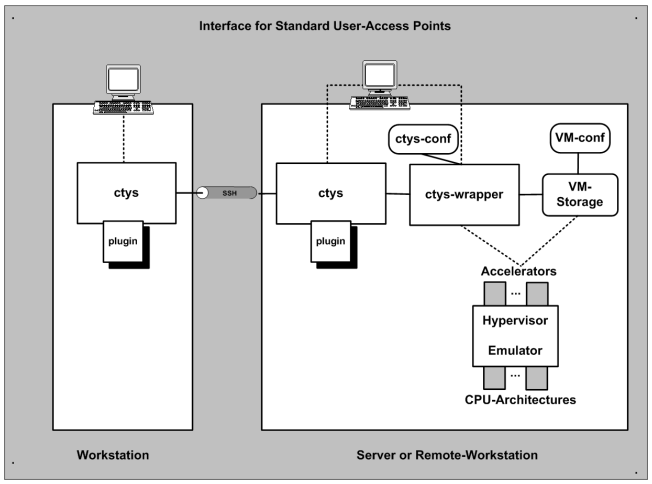

Figure 1: ctys distributed access

The main advance of using a wrapper script is the ability to perform dynamic scripting within the configuration file, which is standard bash-syntax with a few conventions. Templates for configuration files are supported within the **.ctys/ctys-createCofVM.d** directory. The whole set of the UnifiedSessionsManager framework is available within the wrapper scripts.

An installer for complete setup of a QEMU and/or KVM based VM is contained. The tool **ctys-createConfVM(1)** creates either interactively, or in batch-mode a local or remote configuration by detection of the actual platform and creation of a ready-to-use startup configuration. This configuration comprises a generic wrapper script and a specific configuration file. The installation of a GuestOS could be performed either by calling the wrapper-script or by calling ctys with the **BOOTMODE** set to **INSTALL** for ISO image boot, or to **PXE** for network based boot of the install medium. Once QEMU/KVM is setup, the boot of the VM could be performed from the virtual HDD.

Basic Use-Cases for application are contained within the document **ctys-uc-QEMU(7)** .
.

2 Supported HOST-OSs

The QEMU plugin is supported an all released runtime environments of the UnifiedSessionsManager.

3 Supported GuestOSs

The native GuestOS support is the same as for the PMs and HOSTs plugins.

4 Supported Architectures

The whole set of QEMU's CPUs is supported, which includes for version 0.9.1:

`x86, AMD64, ARM, MIPS, PPC, PPC64, SH4, M68K, ALPHA, SPARK`

The call has to be configured within the configuration file. Ready-to-use templates for the provided QEMU tests are included for x86, Arm, Coldfire, and SPARC - Running Linux, uCLinux, and NetBSD.

5 Supported Interfaces

5.1 Overview

Qemu supports various interfaces for interconnection of it's hosted GuestOS to an external devices. Particularly the applicable interfaces for **CONSOLE** and **QEMUMONITOR** interconnection are of interest for the QEMU plugin as a hypervisor controller, whereas the support for native interfaces is handled by the HOSTs plugings.

The encapsulation of the interfaces for access from the **outside-HostOS** to the **inside-GuestOSs** is encaspsulated by the QEMU-VM via usage of specific virtualisation drivers. These drivers actually manipulate the payload-dataflow and are commonly interconnected to native operational peers of the GuestOS such as the LAN interfaces. The outer encapsulation by the UnifiedSessionsManager is a control only encapsulation and interconnects just the few interfaces required for the control of the hypervisor as well as the user interfaces.

Some addional tools are provided as helpers for configuration and management of HostOS operatinal interfaces. One example is here the **ctys-setupVDE(1)** script for the interconnection of the virtual QEMU network interface stubs to their operational HostOS peers.

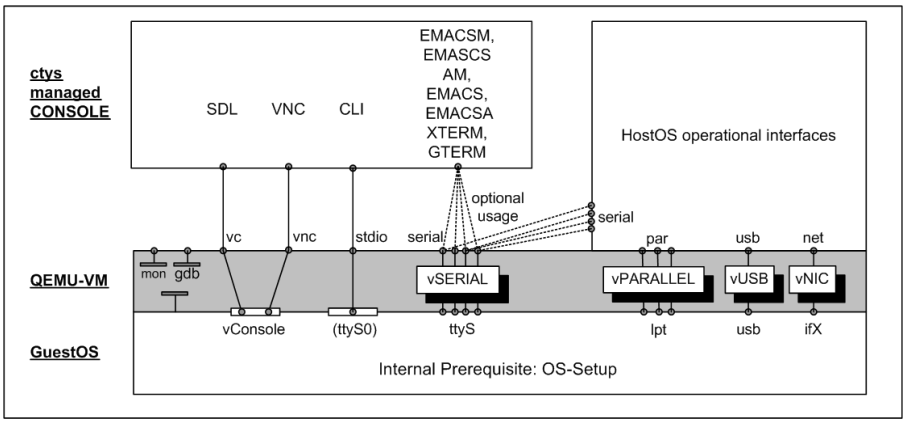

Figure 2: Supported Management CONSOLEs

In the previous layered interface depiction the serial interfaces could be optionally interconnected by CONSOLE entities as well as be used for HostOS devices.

The structure of the encapsulation and the supported components are depicted within the following figure.

The outer encapsulation by the UnifiedSessionsManager is divided into two parts. The first part is the custom wrapper-script for final execution of the QEMU VM by calling the VDE-wrapper. The ctys-wrapper script itself can represent a complex control flow but is mananaged as one entity only, thus not more than one VM instance should be implemented within the wrapper script. The second part of the interface is the call interface for specific CONSOLE types, which prepares additional execution environments for the various call contexts. It should be obvious that the two outer encapsulation components are required to cooperate seamless.

For the actual and final interconnection to the GuestOS there a two basic styles of CONSOLE types:

1. **Transparent standard IO-Devices**
   These are implemented by the virtual device drivers for keyboard, mouse, and display. The standard drivers of the GuestOS handles these transparently as standard user interfaces.

   Transparent IO-Devices are SDL and VNC based on a transient virtual HW, thus almost need no specific configuration for standard devices, but the activation for the QEMU VM.

2. **GuestOS custom IO-Devices**
   These are optional configurations for GuestOSs such as a serial console within Linux as a GuestOS. In case of Linux for example the user has to prepare the usage by a kernel parameter for boot time access and preset a tty-console-device in "/etc/inittab".

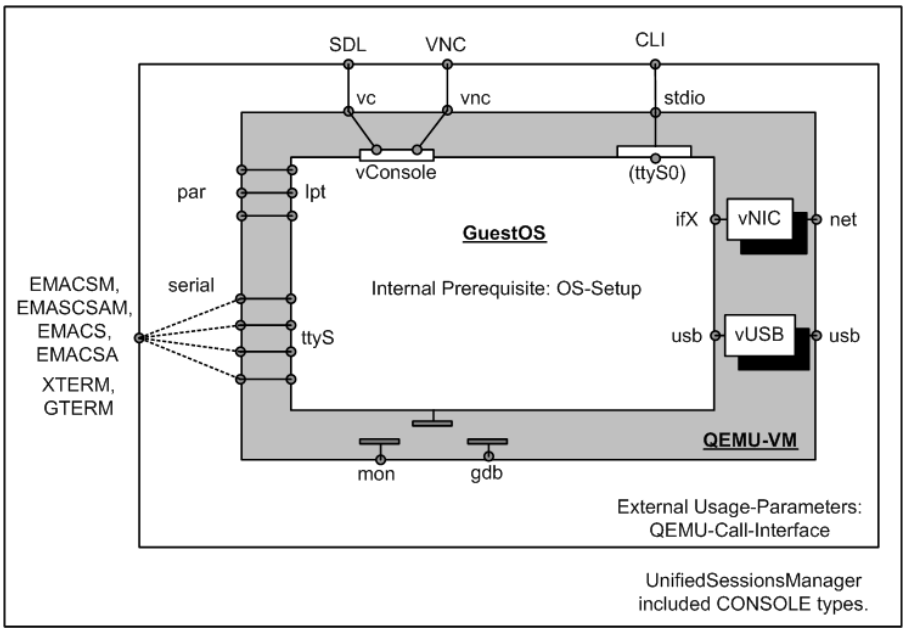

Figure 3: Supported QEMU Management Interconnection-Interfaces

QEMU supports by default up to 4 serial devices. Within the UnifiedSessionsManager, one port is forseen for CLI , VNC , and SDL mode, two serial ports are foreseen for the remaining modes to be used by the framework. For the CLI console no extra monitoring port is allocated, the default values for **-nongraphic**, which are stdout/stdin with a multiplexed monitoring port, are used.

One serial device is reserved for an additional monitor port exclusively for the types SDL and VNC . For the remaining CONSOLE types, which are variants of CLI type, the monitoring port is multiplexed to the console port again, but now for an allocated common UNIX-Domain socket. This port is required in order to open a management interface. When suppressed some actions like CANCEL may not work properly. This is for example the case, for the final close of the stopped QEMU VM, which requires frequently a monitor action.

```
-serial mon:unix:\${MYQEMUMONSOCK},server,nowait
```

5.2 Serial Ports

The setup of a serial console for QEMU is required for various CONSOLE types. Any CONSOLE providing an ASCII-Interface, except the syncronous un-detachable CLI console, requires serial access to the GuestOS. This is a little complicated to setup for the first time, but once performed successful, it becomes an easy task for frequent use.

The first thing to consider is the two step setup, which comprises the initial installation with a standard interface either by usage of SDL or by usage of the VNC console. The second step - after finishing the first successesfully - is to setup the required serial device within the GuestOS. This requires a native login as root. Detailed information is for example available at "Linux Serial Console HOWTO" [6, VANEMERY], "Serial-HOWTO" [7, GHAWKINS], and "Text-Terminal-HOWTO" [8, DSLAWYER].
The following steps are to be applied.

1. Install GuestOS by usage of SDL/ VNC as console.
2. Login into the GuestOS.
3. Adapt **/boot/grub/menu.lst**

   The header section:

   ```
   serial --unit=0 --speed=9600 --word=8 --parity=no --stop=1
   terminal --dumb serial console
   splashimage=(hd0,0)/grub/splash.xpm.gz

   default=<\#yourKernelWithConsole>
   ```

   Your target kernel for boot <yourKernelWithConsole>:

   ```
   kernel ... console=tty0 console=ttyS0,9600n8
   ```

4. Adapt **/etc/inittab**, here for CentOS-5.0

   ```
   S0:12345:respawn:/sbin/agetty ttyS0 9600 Linux
   ```

5. Adapt ctys-qemu-wrapper This requires adapted items, which depend on the choosen CONSOLE type. - A CLI , which is a synchronous console, requires:

   ```
   -nographic
   ```

   This switches off the graphic and defaults it's IO to the caller's CLI .

6. A "by-Window-Encapsulated-CLI", which is a non-"modal" console, called as a serial console by a UNIX-Domain socket within a X11-Window/Client, requires:

   ...

   This switches off the graphic and defaults it's IO to the caller's CLI .

7. Reboot.

5.3 STDIO

This attaches the console to the callers shell. Requires preconfiguration of a serial device within the GuestOS, for a template refer to setup of serial console.

5.4 Network

The prefered network devices are based on the virtual switch provided by the VirtualSquare-VDE project. These are attached to TAP devices with root permissions and require from than on only user permissins for attaching VMs to the virtual switch. Even though any provided network connection could be utilized within the wrapper script, the current toolset supports the VDE utilities only.

The VDE project provides a wrapper for the qemu call, which replaces the qemu call by **vdeqemu**. The parameters "-net nic,macaddr=$MAC0" and "-net vde,sock=$QEMUSOCK" are used within the standard wrapper scripts.

For additional information refer to the chapter "Network Interconnection".
This console replaces the SDL type when choosen. It works as a virtual Keyboard-Video-Mouse console by default and thus does not require pre-configuration of the GuestOS. But needs to be explicitly activated by the **-vnc** option.

5.5 SDL

SDL is the probably intended "standard" device, but has in some versions the drawback of cancelling the VM when the window is closed. Within ctys the safely detachable VNC connection is the preferred console. When for analysis of the boot process the BIOS output is required the CLI console could be applied.

5.6 Parallel Ports

ffs.

5.7 USB

ffs.

5.8 Bluetooth

ffs.

5.9 FDD

ffs.

5.10 HDD

ffs.

5.11 CDROM/DVD

The default wrapper-script contains one HDD as hda device for the **BOOTMODE:VHDD** and additionally one DVD/CDROM for the **BOOTMODE:INSTALL**. These could be extended as required.
For dynamic non-stop-configuration of a DVD/CDROM the following procedure has to be applied within the QEMUmonitor.

1. info block
2. eject <device>
3. change <device> <path-to-iso-file/path-to-dev-cdrom>

6 Supported VM Management Interfaces - QEMUmonitor

The QEMU monitor port is supported as a local UNIX-Domain socket only. The socket name is assembled by a predefined environment variable and the PID of the master process for the final wrapper script , which is executing the QEMU VM and has to be configured by the user. For the various CONSOLE types different handling of the monitor port is applied:

- CLI0: No specific port, stdio is used for console as well as for monitor.
- SDL, VNC : Extra monitor port QEMUMONSOCK, used as multiplexed port, could be connected additionally by an ASC-II console.

CLI , XTERM, GTERM, EMACS, EMACSM, EMACSA, EMACSAM: Mapped monitor port in multiplex mode on UNIX-Domain socket QEMUMONSOCK for re-attacheable console port.
The base variable is "QEMUMONSOCK" , which contains by convention the substrings **ACTUALLABEL** and **ACTUALPID**. These two substrings will be replaced by their actual values evaluated when valid during runtime. The **ACTUALLABEL** is the label of the current VM, as will be provided to the commandline option **-name** of QEMU. The **ACTUALPID** is the master pid of the wrapper script, which will be evaluated by the internal utility "ctys-getMasterPid" . The master pid is displayed as the SPORT value, even though it is used as a part of the actual UNIX domain socket only.
The default socket-path is:

```
/var/tmp/qemumon.<ACTUALLABEL>.<ACTUALPID>.$USER
```

This will be replaced e.g. to:

```
/var/tmp/qemumon.arm-test.4711.tstuser1
```

Any terminal application like **unixterm** of VDE package, or **netcat/nc** could be used for interaction. The switch between QEMU monitor and a text console is the same as for the **-nographic** mode by **Ctrl-a-c**, for additional information refer to the QEMU user-manual. The monitor socket is utilized by internal management calls like CANCEL action by usage of **netcat/nc**.

REMARK: When terminating a CLI session, the prompt will be released by a monitor short-cut: **Ctrl-a x**. In some cases a **Ctrl-c** is sufficient.

The following controls are used for monitor:

Ctrl-a	**001**
x	170
c	143
S3	stop/cont
S4	savevm/loadvm[tagid]
S5	Ctrl-ax

Utilized QEMU-Monitor-Commands

The switch over between the guest console and the monitor console from within a **VNCviewer** client is performed by **Ctrl-Alt-(1|2)**. Where **Ctrl-Alt-2** switches to the Monitor, and **Ctrl-Alt-1** back to the GuestOS-Console. When nested VNCviewers are called the VNCviewer-Menu by default opended with **F8** could be used to mask either the **Ctrl** or the **Alt** key.

7 Network Interconnection

7.1 Overview

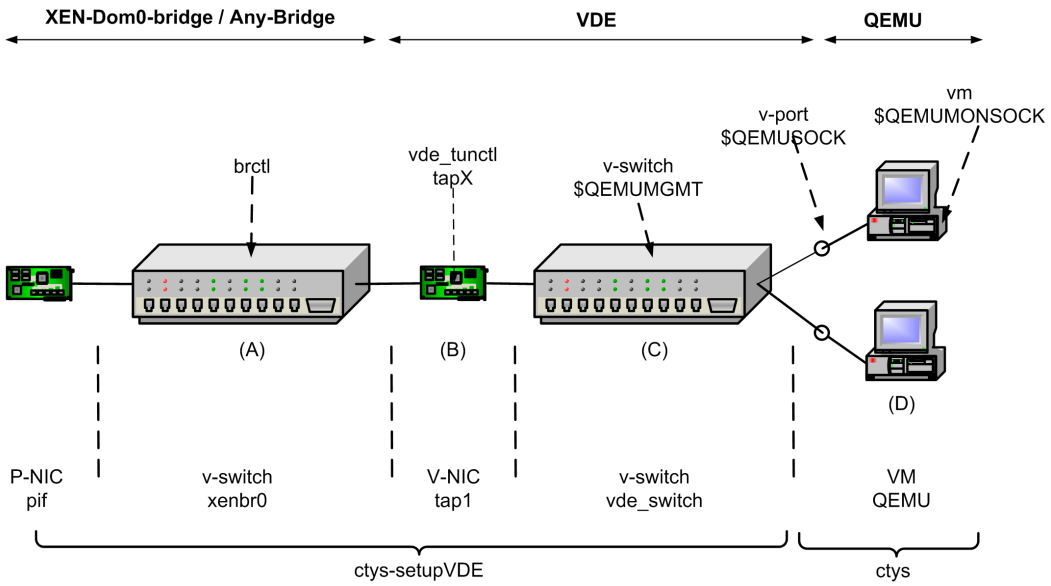

Figure 9: QEMU NIC interconnection

The QEMU plugin utilizes the VDE package exclusively for setting up network connections. The verified version is vde2 which is for the current version of ctys-QEMU a mandatory prerequisite. This is due to the following two features mainly:

- Availability of tunctl, which is named **vde_tunctl** within vde. A TAP device is mandatory for QEMU to be interconnected in a transparent bridging mode as applied here.
- The User-Space virtual switch **vde_switch** requires one tap-device to attach itself to the **external** network, whereas the users can attach to it's internal ports with their own permissions. This is permitted due to pre-assignment of it's owner by usage of **vde_tunctl** .

A short description of the install process for QEMU with network support, pxe-boot/install, and cdrom-boot/support based on the examples from QEMU is given in the HowTo of ctys. Downloads are available from [9, VDE2], and a very good decscription about networking with TAP could be read at [10, VirtualSquare].

Once the basic install and setup is completed, the whole process for the creation of the required networking environment is handled for local and remote setups by one call of the **ctys-setupVDE(1)** only .

The listed environment variables are to be used within the configuration scripts. These are particularly mandatory for to be present and accessible by usage of ctys. So the CANCEL action for example will open a connection to the QEMUMONSOCK and sends some monitor commands. The QEMUMGMT variable will be used to evaluate the related tap-device, and for final deletion of the swithc, when no more clients are present. All sockets are foreseen to be within UNIX domain only, as designed into the overall security principle. Anyhow some minor break might occur for the vnc port for now, and should be blocked by aditional firewall rules for remote access.

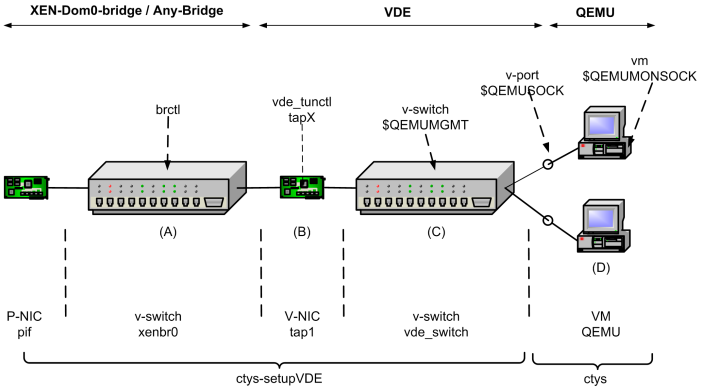

Figure 10: QEMU interconnection

The following variables are required partly to be modified with dynamic runtime data such as the actual USER id and the PID as shown in the examples. The definition and initialization is set in the central plugin-configuration file **qemu.conf**.

- QEMUMONSOCK The monitoring socket within local UNIX domain, will be used as

  ```
  -serial mon:unix\${QEMUMONSOCK},server,nowait
  ```

  Could be attached by the terminal emulations. It should be the first entry containing the serial console "ttyS0" too, which is for the provided ctys-examples assumed, and is the case.

  - nc -U $QEMUMONSOCK
  - unixterm $QEMUMONSOCK

- QEMUSOCK The socket to be attached to the vde_switch. One specific port, therefore specific QEMUSOCK is derived for each running QEMU instance.

  The network socket within local UNIX domain, will be used as

  ```
  -net vde,sock=\${QEMUSOCK}.
  ```

- QEMUMGMT The socket for management of the virtual switch. This could be also accessed by nc/netcat and unixterm. The utility **ctys-setupVDE(1)** widely utilizes this interface.

  - nc -U $QEMUMGMT or netcat -U $QEMUMGMT
  - unixterm $QEMUMGMT

7.2 Setup Networking for QEMU by VDE

Once QEMU and VDE2 are istalled successfully, either by delivered packages or by compilation and the "make install" call, the base package of QEMU is installed. The next step now is to create a runtime environment.

The current version therefore supports particularly the tools **ctys-setupVDE(1)** and **ctys-createConfVM(1)**

- **ctys-createConfVM(1)**
  handles the whole process of required preparation for the final GuestOS installation. Therefore several configuration and wrapper scripts are created. These comprise particularly ready-to-use installer executables in case of debian-5.x or CentOS-5.x as GuestOS.

- **ctys-setupVDE(1)**
  handles the complete process of creation and interconnection of a virtual switch by just one call.

The QEMU project offers some ready-to-use images, which could be used instead of the creation of a new VM.

- arm-test-0.2.tgz

- coldfire-test-0.1.tar.bz2
- linux-0.2.img.bz2
- mipsel-test-0.2.tgz
- mips-test-0.2.tgz
- small.ffs.bz2
- sparc-test-0.2.tgz

The only configuration required later is the setting of appropriate IP address, except for the coldfire image, which is based on DHCP. This is only true if you are using DHCP and have set the appropriate MAC addresses.

REMARK: The usage of bridged network with communications via the NIC of the host requires some additional effort. Particularly the creation of the required TAN-device with the frequently mentioned **tunctl** utility from the **UserModeMLInux** was somewhat difficult on CentOS-5.0. The package **vde** including a (not-found-documentation-for) utility **vde_tunctl** was the rescue-belt. STarting with the first version this is completely encapsulated by the utility **ctys-setupVDE(1)**

The resulting call to setup a compelte networking environment is

```
ctys-setupVDE -u <userName> create
```

Some deviation may occur in case of multiple interfaces, where the first is not active. In such cases it is sufficient to provide the option '-i' for selection of a specific interface.
At this point anything might be prepared for successful operations and the installation of a GuestOS could be performed as described within the following chapters.

For Qemu several excellent sites with install descriptions exist, thus here are just some shortcuts, which seem to be the most important items.

- **PXE-Boot**
  The following applies to a configured PXE environment based on PXELinux, check this with the call option '-d pf'.

```
vdeqemu -vnc :17 -k de -m 512
        -hda linux-0.2.img
        -net nic,macaddr=<mac>
        -net vde,sock={QEMUSOCK}
        -boot n
        -option-rom \${QEMUBIOS}/pxe-ne2k\_pci.bin \&
```

- **ISO-Image-Boot**
  The following applies to a boot CDROM image for installation, check this with the call option '-d pf'.

```
vdeqemu -vnc :17 -k de -m 512
        -hda linux-0.2.img
        -net nic,macaddr=<mac>
        -net vde,sock={QEMUSOCK}
        -boot d
        -cdrom \${QEMUBASE}/iso/install.iso \&
```

- **shared memory**
  Should be set in **/etc/fstab** as required. This value might be raised, when nested stacks are used, thus the bottom engine requires the sum of the resources of the stack. The entry might look like:

```
none /dev/shm tmpfs defaults,size=512M 0 0
```

For a call like:

```
vdeqemu -m 512 ...
```

The changes could be activated with

```
mount -o remount /dev/shm
```

- **Create image**
  To create an ISO image a call like the following could be applied

```
qemu-img create -f qcow myImage.qcow 4G
```

which could be used as

```
qemu -cdrom installMedia.iso \
  -boot d myImage.qcow
```

7.3 PXE-Boot

The PXE based installation is possibly not the fastest, but it offers a common seamless solution for unified installation processes. Even though an image could be just copied and modified as required, some custom install procedures might be appreciated, when the install could be performed in batch-mode. One example is the usage of kickstart files for CentOS/RHEL. In case of PXE these files are almost the same for any install base, this spans from physical to virtual machines.

7.4 Install on USB-Sticks

7.4.1 FreeDOS - Balder for BIOS-Updates

The installation of FreeDOS on a bootable USB-Stick e.g. for BIOS updates requires the following steps.

1. Create configurationfiles and wrapper script by calling **ctys-createConfVM(1)** .
2. Executes first stage of installation including formatting of required boot device by calling

```
ctys -t qemu -a create=l:myLabel,instmode:FDD\%none\%USB\%/dev/sdg\%init localhost
```

This call requires the configuration of the path to the FDD image within the configuration file, thus 'none' is provided within the call.

The installation requires the following actions by the user within the installed system.

1. Call of **fdisk** in order to partition the target device for installation.
2. Reboot.
3. Execute second stage of installation including the format of device by calling.

```
ctys -t qemu -a create=l:myLabel,instmode:FDD\%none\%USB\%/dev/sdg\%none localhost
```

This call requires the configuration of the path to the FDD too and omits the initial formatting by setting the stage to none.

The following actions by the user are quired.

4. Call of **format c: /S** in order to format the target device.
5. Call of **xcopy /E /N a: c:** for copy of executables.
6. Adaption of autoexec.bat.

That's it.

7.4.2 Linux - CentOS

The installation of Linux is even easier. Just call the installmode and assign the path to the USB device as inst-target. The installer reqcognizes the USB device and handles the partitioning.
.

8 Installation of Components

8.1 Install UnifiedSessionsManager

This is assumed to be done already when reading this document, else refer to the release notes.

8.2 Install VDE2-2.2.3 from Sources

1. Download and install source.
2. Configure

```
./configure --prefix=/opt/vde-2.2.3
```

3. make
4. set symbolic links

```
ln -s /opt/qemu-0.12.2 /opt/qemu
```

5. Verify installation by creation of a switch, requires root permissions.

```
ctys-setupVDE -u <user-switch> create
```

8.3 Install KVM as rpm

CentOS-5.4: The installation of the RPM package is quite forward. For KVM on CentOS-5.4 install the whole virtualization group. The QEMU package is installed here from the source.

Fedora: For Fedora the same procedure.

8.4 Install QEMU-0.12.2 from Sources

CentOS-5.4

1. Download and install sources.
2. Install gcc.
3. Configure

```
./configure --prefix=/opt/qemu-0.12.2
```

4. make
5. make install
6. set symbolic links

```
ln -s /opt/qemu-0.12.2 /opt/qemu
```

7. Verify installation

```
ctys-plugins -T QEMU,VNC,X11,CLI -E
```

When the last error message complains the absence of QEMUSOCK and QEMUMGMT, than anything seems to be perfect, just missing the final call for network setup by

8. Call "ctys-setupVDE", requires root permissions.

```
ctys-setupVDE -u <user-switch> create
```

9. Verify installation

```
ctys-plugins -T QEMU,VNC,X11,CLI -E
```

This should work now.

9 Install Procedures

9.1 Install with Manual Setup

This is just depicted for basic demonstration purposes only, thus presents a minimalistic approach by avoidance of enhanced settings such as network devices.

1. Create directory:

```
mkdir myLabel && cd myLabel
```

2. Create a Disk:

```
qemu-img create -f qcow2 disk.img 5G
```

3. Initialize disk image;

```
dd if=/dev/zero of=disk.img bs=1G count=5
```

4. Install from CDROM iso image:

```
qemu-system-x86_64 -hda disk.img \
  -cdrom ${PATHTOIMG}/CentOS-5.4-x86_64-bin-DVD.iso \
  -boot d
```

5. Boot form Disk-Image into GuestOS:

```
qemu-system-x86_64 -hda disk.img
```

9.2 Install with ctys-createConfVM

9.2.1 Install the same GuestOS as the HostOS

The following workflow is foreseen to setup both, pure QEMU based machines and KVM based VMs. The variants KVM as the KQEMU are parameterized accelerators only, which fit perfectly into the overall concept of QEMU. The given example is processed on a HOST machine running CentOS-5.4 and installs the same a GuestOS. The recipe is straight forward and avoids extended details and options for simplicity, e.g. the GuestOS is by default set to the same as the HOST OS.

In case of errors during the start of the VM after the creation the first file to be checked is the file named '<label>.ctys'. This file contains the configuration settings related to the emulated hardware of the VM. Particularly the type of network interface card and the graphics card may cause problems and require to be changed for some GuestOS when appropriate drivers are not available.

1. Change to the HOST machine where the VM is executed. The whole procedure could be executed remote to, but for first trial the local execution avoids some details with required extended knowledge.

2. For the first time execution check the state of the installed components. The call for display of the state with required subcomponents is:

```
ctys-plugins -T qemu -E
```

   The result should be in 'green' state. When errors related to QEMUSOCK and/or QEMUMGMT occur the utility 'ctys-setupVDE' is required to be executed. Due to the allocation of a TAP/TUN device this requires root permissions.

   When errors occur it could be helpful to check the actual required system components. This could be performed by the call:

```
ctys-plugins -d 64,p -T qemu -E
```

3. Create a directory for storage of the set of files comprising the virtual machine and change into.

4. Install on a machine locally

   The present example is foreseen for interactive execution. Several of the interactively polled values could be pre-set by environment variables, the available variables and current default values are shown by the '–list-env-var-options' option(shortcut '–levo'). Additionally init files could be used to set common defaults. The whole procedure could be additionally performed either semi or fully automatic, which requires the whole set of values to be pre-set appropriately.

```
ctys-createConfVM -t QEMU --label=tst253
```

   You will be asked several questions related to the new VM. The resulting configuration is displayed and stored to a configuration file within the machines directory.

   The parameters LABEL and MAC address are particularly important, because these define per convention the network access facilities of the VM. The MAC address is sufficient when DHCP is configured, else the TCP/IP address has to be set manually. For automatic consistency checks of the MAC and TCP/IP address match a '/etc/ethers' alike database could be setup either by ctys-extractMAC or by ctys-extractARP.

5. Check the contents of the configuration file and wrapper-script for the VM.

6. If not yet done, create a virtual switch, refer to 'ctys-setupVDE' for automation.

```
ctys-setupVDE -u <switchuser> create root@app2
```

   When executed remotely from a mounted filesystem this could disconnect in some cases the machine. This is due to the required reconfiguration of some network devices, where some of the re-establishment tools may be stored within the disconnedted filesystem. If this occurs use a local account with locally stored ctys files.

7. Start iterative checks by calling the created wrapper script from the commandline. This is shown here for demonstration purposes of the interface only. The actually required set of checks are performed silently by the QEMU/KVM plugin for verification before each call.

```
sh yourWrappername.sh --print --check
```

   The following initial errors may occur:

   (a) ERROR:Missing QEMUSOCK
   SOLUTION: Call ctys-setupVDE

(b) ERROR: Missing boot image
SOLUTION:Set option "–bootmode=INSTALL"

(c) ERROR: Missing INSTCDROM
SOLUTION: Edit config file or use PXE.

(d) ERROR: Unknown display CONSOLE
SOLUTION: Set option "–console=VNC"

(e) ERROR: Missing VNCDISPLAY
SOLUTION: Set option "–vncaccesdisplay=77"
Here choosen "77" for VNC display.

(f) When now a final assembled call is displayed anything should be fine.

(g) Start the call with removed "–check" option.

(h) ERROR: TUNGETIFF and TUNSETSNDBUF
SOLUTION: Ignore these messages.

8. Start installation with the created wrapper script from the local commandline of the HOST machine.

```
sh yourWrappername.sh --console=vnc --vncaccessdisplay=77 --instmode --print
```

The important parameter is here '–instmode', either with or without suboptions. When missing this option the HDD is used as boot device and fails due to missing OS - of course, it is not yet installed.

Alternatively the installation could be performed by calling ctys with the INSTMODE suboption, which is forseen as the standard operation. The call could be the following from within the VMs subdirectory, when the defaults are used for INSTMODE:

```
ctys -t qemu -a create=ID:\$PWD/yourWrappername.ctys,INSTMODE'
```

When alteration of the INSTMODE suboptions is required the following could be applied:

```
ctys \
  -t qemu \
  -a create=ID:\$PWD/yourWrappername.ctys,INSTMODE:CD\%default\%VHDD\%default\%INIT
```

The warnings related to deprecated support of 'vdeq' could be ignored for curent version. The errors related to TUNGETIFF and TUNSETSNDBUF too.

The release CentOS-5.5 requires the change of the DVD medium during installation. Therefore with 'Ctrl-Alt-2' the user interface could be changed to the monitor terminal, where the commands as described in the chapter about CDROM/DVD could be utilized.

The current machine could be canceled either within the monitor or by the call:

```
ctys -t qemu -a cancel=id:\$PWD/yourWrappername.ctys,poweroff:0,force
```

9. Attach a console by usage of "vncviewer :77&", which could require a short startup timeout of the VM before the execution of the vncviewer. Now proceed with the native GuestOS installation.

10. After installation confirm the reboot of the GuestOS, but when the machine hangs just stop it and restart with "–bootmode=VHDD".

11. Once rebooted finish the post-install steps of the GuestOS.

12. That's it.

9.2.2 Install a GuestOS different from the HostOS

The following workflow is quite similar to the previous case, where the GuestOS is identical to the HostOS. There are just a few deviations for the call of ctys-createConfVM, which requires the GuestOS now to be set either manually, or for frequent usage by a configuration file. The settings could be checked by the option '–levo'.

```
OS=Linux \
OSREL=2.6.26-1-amd64 \
DIST=debian \
DISTREL=5.0.0 \
MAC=00:50:56:16:11:0b \
ACCELERATOR=KVM  \
ctys-createConfVM \
   -t QEMU \
   --label=inst010
   --auto-all
```

When all defaults are pre-set in configuration files the option '–auto-all' could be used as given. The creation of the whole set of initial files than requires about 2-4seconds!

9.3 PXE-Boot

The PXE boot in the current versions of QEMU work on CentOS-5.4 from the box. Just start the VM by calling the wrapper script with the option "–bootmode=PXE" or use the "BOOTMODE:PXE" suboption for the create action of ctys.

9.4 ISO-Image and DHCP

The installation from an ISO image reuqires the fully qualified absolute pathname to be set within the conf-file. Additonally the option "–bootmode=INSTALL" or the suboption "BOOTMODE:INSTALL" for ctys is required.

9.5 Supported/Tested Install-Mechanisms

The actual applicable install mechanism partly depends on the host system, e.g. debootstrap is currently available on debian hosts only.

OS	deboostrap	KSX	PXE	CD	FD	HD	USB
Android-2.2	-			*)			
CentOS-5.0	-	X	X	X			
CentOS-5.4	-	X	X	X			
CentOS-5.5	-	*)	*)	X			
Debian-4.0r3-ARM	-	-		X			
Debian-4.0r3		-	X	X			
Debian-5.0.0		-	X	X			
Debian-5.0.6		-	*)	X			
Fedora 8	-	X	X	X			
Fedora 10	-	X	X	X			
Fedora 12	-	X	X	X			
Fedora 13	-	*)	*)	X			
FreeBSD-7.1	-	-		X			
FreeBSD-8.0	-	-		X			
FreeDOS/balder	-	-	-		X		
Mandriva-2010	-	-		X			
MeeGo-1.0	-	-		*)			
ScientificLinux-5.4.1	-	X	X	X			
OpenBSD-4.0	-	-	X				
OpenBSD-4.3	-	-	X				
OpenBSD-4.6(NOK)	-	-	X				
OpenBSD-4.7	-	-	*)				
OpenSuSE-10.2	-	-	X	X			
OpenSuSE-11.1	-	-		X			
OpenSuSE-11.2	-	-		X			
OpenSuSE-11.3	-	-		X			
Solaris 10	-	-		X			
OpenSolaris 2009.6	-	-		X			
Ubuntu-8.04-D		-		X			
Ubuntu-9.10-D		-		X			
Ubuntu-10.10-D		-		X			
uClinux-arm9	-	-		-			
uClinux-coldfire	-	-		-			

Table 1: Supported/Tested Install-Mechanisms

*) Under Test.

.

10 Installation of Guests

10.1 Android

Refer to the specific Use-Case **ctys-uc-Android** .

10.2 CentOS-5

The given example is based on the following configuration:

- Host executing *ctys-createConfVM(1)* on debian-5.0.0, no qemu installation.
- Target host for execution of QEMU-KVM with OpenSUSE-11.2, present qemu and kvm installation.
- GuestOS within the KVM based VM is CentOS (*ctys-uc-CentOS(7)*) .

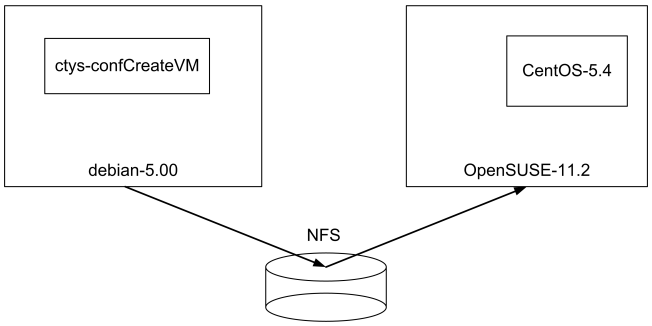

Figure 40: Distributed installation by 'ctys-confCreateVM'

The 'cross-installation' for a different machine requires various options to be set for the target execution environment of the hypervisor which cannot be detected automatically on the local machine. One example is the actual support for the architecture.

Several of the parameters has to be set by environment VARIABLE based options. The available options could be visualized by the call option '–list-environment-variable-options' or for short '-levo'. The initial default values wre displayed too, which is not available for dependent values. E.g. by convention wihtin ctys the LABEL of a VM is the hostname of the conatined GuestOS, therefore the value of TCP/IP parameters could be determined only after the LABEL value is defined. The options by environment variable are applicable for local execution only, else the interactive dialogue is available only. The most value of the pre-defined values is gained in combination with oneof the options '–auto' or '–auto-all'. These pre-define confirmation answers for the interactive dialogues.

The following call creates in place of appropriate default values fully automated the complete set of configuration files and wrapper-scripts, including a basic kickstart file.

```
ACCELERATOR=KVM \
DIST=CentOS  \
RELEASE=5.4 \
OS=Linux  \
OSVERSION=2.6.18 \
ctys-createConfVM \
  -t QEMU \
  --label=tst488 \
  --no-create-image \
  --auto-all
```

When dropping the '–auto-all' option an interactive dialogue is performed, where no default values are required neccessarily. The '–no-create-image' avoids the attempt to create a new image, this could be helpful when a present image should be just re-configured, e.g. for usage of an alternative ACCELERATOR, which could be altered by configuration file only. This could be done manually by editing the configuration file too, of course.

The start of the VM requires the execution of 'ctys-setupVDE' for creation of a virtual bridge and a virtual switch. The prerequisite is the complete installation of 'qemu' and 'vde', current tested version for vde is 'vde2-2.2.3' which is available from source-forge. The 'prefix' option should be set by the required 'configure' call to '/opt'. After installation of the sources of vde2 the following should be performed within the installation directory.

```
make clean
./configure --prefix=/opt/vde2-2.2.3
make
make install
ln -s /opt/vde2-2.2.3 /opt/vde
```

The virtual switch could be created by the following call.

```
ctys-setupVDE -u tstUser create
```

A start of the VM is typicall called by the following command:

```
ctys -t QEMU -a create=l:tst488,b:<base-path-VM>,reuse tstUser@testHost
```

10.3 Debian-5

The installation of debian is straight forward in accordance to the generic description for CentOS. The following pitfalls have to be avoided when more than one version of QEMU/KVM is installed.

1. The variable QEMUBIOS has to point to the suitable set of BIOS modules for the actually executed version. By default the omission of the variable at all should provide that. The setting could be handeled by SHELL, config-QEMU-files, and within the specific VM.
2. Probably it is a good idea to deactivate the screensaver first, the screen saver for the HOST should suffice.

The basic setup of debian-lenny could be performed with following steps:

1. Install debian-lenny-5.0.0/5.x with basic settings, here Gnome.
2. Add: tightVNC server and client
3. Add: sshd
4. Add: bridge-utils
5. Add: socat (required for '-U'-UnixDomain-Option).
6. Configure password-less login for SSH e.g. ssh-copy-id or by Kerberos.
7. Install ctys e.g.:

   ```
   ctys-distribute -F 1 -P UserHomeCopy root@tst210
   ```

   For activation of environment variables either a fresh login or the manual 'source' of the '$HOME/.bashrc' or the '$HOME/.profile' is required on the target machine.
8. Remote execution of eg.

   ```
   ctys -a info root@tst210
   ```

9. Remote or local execution for preparation of the inventory data and the normalized performance characteristic for possible comparison by:

   ```
   ctys-genmconf -P -x VM root@tst210
   ```

10. Check now results of remote execution for PM-section

    ```
    ctys -a info root@tst210
    ```

11. Start a VNC session

    ```
    ctys -a create=l:ROOT,reuse root@tst210
    ```

10.4 Fedora-10

The installation is quite forward.
Install the base system as described in the generic 'Installation Procedure'. Follow these steps for a base system with hosts and PMs features.

1. Adapt '/etc/yum.repo.d/fedora.repo' appropriately.
2. Install - if no else is available - from the CentOS-5.4 package, use '–nodeps':
   - libsysfs-2.0.0-6.x86_64.rpm
   - vnc-4.1.2.14.el5_3.1.x86_64.rpm
   - vnc-server-4.1.2.14.el5_3.1.x86_64.rpm
   - bridge-utils-1.1-2.x86_64.rpm
3. Configure password-less login for SSH e.g. ssh-copy-id or by Kerberos.
4. Install ctys e.g.:

   ```
   ctys-distribute -F 1 -P UserHomeCopy root@tst240
   ```

   For activation of environment variables either a fresh login or the manual 'source' of the '$HOME/.bashrc' or the '$HOME/.profile' is required on the target machine.
5. Remote execution of eg.

   ```
   ctys -a info root@tst240
   ```

6. Remote or local execution for preparation of the inventory data and the normalized performance characteristic for possible comparison by:

   ```
   ctys-genmconf -P -x VM root@tst240
   ```

7. Check now results of remote execution for PM-section

   ```
   ctys -a info root@tst240
   ```

8. Start a VNC session

   ```
   ctys -a create=1:ROOT,reuse root@tst240
   ```

10.5 FreeBSD-7.1

1. Install by DVD-image
2. Add: tightVNC, bash, xorg, pciutils, gnome-sessions, fvwm

10.6 FreeBSD-8.0

1. ctys-createConfVM
2. Install by DVD-image
3. Configure network with DHCP, and ssh-login
4. chpass for setting bash
5. Add tightVNC and pci-utils
   ffs.

10.7 MeeGo

Refer to the specific Use-Case **ctys-uc-MeeGo** .

10.8 OpenBSD-4.3+SerialConsole - by PXE

The installation is straight forward with BOOTMODE=PXE. When for later access a serial console is required following additional steps has to be proceeded. A serial console is a prerequisite for EMACS, XTERM, and GTERM.

1. Boot and login into GuestOS if the root-filesystem cannot be mounted offline.
2. Create a file "/etc/boot.conf" with content such as:

```
set tty com0
stty com0 115200
set timeout 15
boot
```

+Edit the file "/etc/tty" and change the lines:

```
tty00   "/usr/libexec/getty std.115200" vt220 on secure
```

10.9 OpenBSD-4.6+SerialConsole - by PXE

The Version of kvm-83 on CentOS-5.4 requires following steps for boot of installation:

1. Use e1000 driver for NIC.
2. Boot by calling: 'bsd.rd -c'
3. Disable mpbios:'disable mpbios'

After install make the changes persistent with:

1. config -f -e /bsd
   - disable mpbios
   - quit
     The following packages are required in addition to the base installation.
2. tightVNC + tightVNCviewer

3. bash
4. pciutils
5. eventually gnome-session

   Once the installation is complete the following steps should be proceeded:
6. Aktivate X11Forwarding for SSH
7. Execute:

   ```
   ctys -a info root@host
   ```
8. Execute:

   ```
   ctys-genmconf -P -x VM root@host
   ```

10.10 OpenSolaris-2009.6

ffs.

10.11 openSuSE-11.2

Install the base system as described in the generic 'Installation Procedure'. Follow these steps for a base system with hosts and PMs features.

1. Add your repository within yast and install.
   - bridge-utils
2. Download vde - verified with vde2-2.2.3

   [sourceforge.net/projects/vde sourceforge.net/projects/vde]
   - install sources
   - call 'configure –prefix=/opt/vde2-2.2.3'
   - call 'make'
   - call 'make install'
   - call 'ln -s /opt/vde2-2.2.3 /opt/vde'
3. Configure password-less login for SSH e.g. ssh-copy-id or by Kerberos.
4. Install ctys e.g.:

   ```
   ctys-distribute -F 2 -P UserHomeCopy root@tst214
   ```

   For activation of environment variables either a fresh login or the manual 'source' of the '$HOME/.bashrc' or the '$HOME/.profile' is required on the target machine.
5. Remote execution of eg.

   ```
   ctys -a info root@tst214
   ```
6. Remote or local execution for preparation of the inventory data and the normalized performance characteristic for possible comparison by:

   ```
   ctys-genmconf -P -x VM root@tst214
   ```

7. Check now results of remote execution for PM-section

```
ctys -a info root@tst214
```

8. Start a VNC session

```
ctys -a create=l:ROOT,reuse root@tst214
```

10.12 ScientificLinux-5.4.1

Install the base system as described in the generic 'Installation Procedure'. Follow these steps for a base system with hosts and PMs features.

1. Add your repository within yast and install.
   - bridge-utils
   - vnc
2. Configure password-less login for SSH e.g. ssh-copy-id or by Kerberos.
3. Install ctys e.g.:

```
ctys-distribute -F 2 -P UserHomeCopy root@tst213
```

   For activation of environment variables either a fresh login or the manual 'source' of the '$HOME/.bashrc' or the '$HOME/.profile' is required on the target machine.

4. Remote execution of eg.

```
ctys -a info root@tst213
```

5. Remote or local execution for preparation of the inventory data and the normalized performance characteristic for possible comparison by:

```
ctys-genmconf -P -x VM root@tst213
```

6. Check now results of remote execution for PM-section

```
ctys -a info root@tst213
```

7. Start a VNC session

```
ctys -a create=l:ROOT,reuse root@tst213
```

10.13 Solaris-10

ffs.

10.14 Ubuntu-8.04

Install the base system as described in the generic 'Installation Procedure'. Follow these steps for a base system with hosts and PMs features.

1. Add your repository within yast and install. Here the debian distribution debian-5.0.0-lenny is used partly.
   - bridge-utils
   - openssh-server
   - tightvnc client+server
2. Download vde - verified with vde2-2.2.3
   [sourceforge.net/projects/vde sourceforge.net/projects/vde]
   - install sources
   - call 'configure –prefix=/opt/vde2-2.2.3'
   - call 'make'
   - call 'make install'
   - call 'ln -s /opt/vde2-2.2.3 /opt/vde'
3. Configure password-less login for SSH e.g. ssh-copy-id or by Kerberos.
4. Install ctys e.g.:

   ```
   ctys-distribute -F 2 -P UserHomeCopy root@tst236
   ```

   For activation of environment variables either a fresh login or the manual 'source' of the '$HOME/.bashrc' or the '$HOME/.profile' is required on the target machine.
5. Remote execution of eg.

   ```
   ctys -a info root@tst236
   ```

6. Login as e.g. tst@tst236

   ```
   ssh -X tst@tst236
   ```

7. Local execution for preparation of the inventory data and the normalized performance characteristic for possible comparison by:

   ```
   sudo PATH=\$PATH:\$HOME/bin ctys-genmconf -P -x VM root@tst236
   ```

8. Check now results of remote execution for PM-section

   ```
   ctys -a info root@tst236
   ```

9. Start a VNC session

   ```
   ctys -a create=1:ROOT,reuse root@tst236
   ```

10.15 Ubuntu-9.10

Install the base system as described in the generic 'Installation Procedure'. Follow these steps for a base system with hosts and PMs features.

1. Add your repository within yast and install. Here the debian distribution debian-5.0.0-lenny is used partly.
   - bridge-utils
   - tightvnc client+server
2. For openssh-server the appropriate Ubuntu package is required due to version dependency.
   - Download package openssh-server
   - install: dpkt -i openssh-server...
3. Download vde - verified with vde2-2.2.3
   [sourceforge.net/projects/vde sourceforge.net/projects/vde]
   - install sources
   - call 'configure –prefix=/opt/vde2-2.2.3'
   - call 'make'
   - call 'make install'
   - call 'ln -s /opt/vde2-2.2.3 /opt/vde'
4. Configure password-less login for SSH e.g. ssh-copy-id or by Kerberos.
5. Install ctys e.g.:

   ```
   ctys-distribute -F 2 -P UserHomeCopy root@tst215
   ```

   For activation of environment variables either a fresh login or the manual 'source' of the '$HOME/.bashrc' or the '$HOME/.profile' is required on the target machine.
6. Remote execution of eg.

   ```
   ctys -a info root@tst215
   ```

7. Login as e.g. tst@tst215

   ```
   ssh -X tst@tst215
   ```

8. Local execution for preparation of the inventory data and the normalized performance characteristic for possible comparison by:

   ```
   sudo PATH=\$PATH:\$HOME/bin ctys-genmconf -P -x VM root@tst215
   ```

9. Check now results of remote execution for PM-section

   ```
   ctys -a info root@tst215
   ```

10. Start a VNC session

    ```
    ctys -a create=l:ROOT,reuse root@tst215
    ```

10.16 QNX-6.4.0

The installation is quite straight forward. The setup of the configuration file and the preparation of the install media could be performed by the following call on the target machine:

1. mkdir <directory>
2. Call this on target machine, due to pre-set environment variables for the first step of installation. This prepares the machine for the actuall installation:

```
INSTCDROM=/mntn/swpool/miscOS/QNX/6.4.0/raw/qnxsdp-6.4.0-200810211530-dvd.iso \
MAC=00:50:56:13:13:18 \
IP=172.20.6.23 \
DIST=QNX-SDP \
DISTREL=6.4.0 \
OS=QNX \
OSREL=6.2.3 \
ctys-createConfVM \
  -t qemu \
  --label=tst323 \
  --auto-all
```

3. Start installation from anywhere:

```
ctys -t qemu \
  -a create=l:tst323,reuse,instmode:CD\%default\%HDD\%default\%init,\
     b:/mntn/vmpool/vmpool05/qemu/test/tst-ctys/tst323 \
  -c local app1
```

4. Cancel installed machine for reboot.

```
ctys -t qemu -a cancel=l:tst323,poweroff root@lab02
```

5. Start QNX.

```
ctys -t qemu \
  -a create=l:tst323,reuse,b:/mntn/vmpool/vmpool05/qemu/test/tst-ctys/tst323 \
  -c local app1

\ \\
```

10.17 QEMU-arm-test

The configuration and integration of the provided test image for ARM based uCLinux is is quite straight forward. The setup of the configuration file and the preparation of the install media could be performed by the following call on the target machine:

1. mkdir <directory>
2. Copy and unpack the image-archive into the fresh directory.

3. Call this on target machine, due to pre-set environment variables for the first step of installation. This prepares the machine for the actuall installation:

```
ACCELERATOR=QEMU \
STARTERCALL=/opt/qemu/bin/qemu-system-arm \
ARCH=arm926 \
NETMASK=255.255.0.0 \
MAC=00:50:56:13:13:19 \
IP=172.20.6.24 \
DIST=QEMU-arm-test \
DISTREL=0.9.1-0.2 \
OS=ucLinux \
OSREL=2.x \
MEMSIZE=128 \
HDDBOOTIMAGE\_INST\_SIZE=2G \
HDDBOOTIMAGE\_INST\_BLOCKCOUNT=8 \
INST\_KERNEL=zImage.integrator \
INST\_INITRD=arm\_root.img \
ctys-createConfVM \
  -t QEMU \
  --label=tst324 \
  --auto-all
```

4. Either edit the sections or just append the following to the 'tst324.ctys' configuration file, refer to provided README:

```
KERNELIMAGE=zImage.integrator
INITRDIMAGE=arm\_root.img

\#For: -nographic
\#APPEND=\${APPEND:-console=ttyAMA0}

CPU=arm926
```

5. Start virtual machine:

```
ctys -t qemu \
  -a create=l:tst324,reuse\
     b:/mntn/vmpool/vmpool05/qemu/test/tst-ctys/tst324 \
  -c local app1
```

6. Cancel installed machine for reboot.

```
ctys -t qemu -a cancel=l:tst324,poweroff root@app1
```

10.18 QEMU-coldfire-test

The configuration and integration of the provided test image for Coldfire based uCLinux is is quite straight forward. The setup of the configuration file and the preparation of the install media could be performed by the following call on the target machine:

1. mkdir <directory>
2. Copy and unpack the image-archive into the fresh directory.
3. Call this on target machine, due to pre-set environment variables for the first step of installation. This prepares the machine for the actuall installation:

```
ACCELERATOR=QEMU \
STARTERCALL=/opt/qemu/bin/qemu-system-m68k \
ARCH=ColdFire \
NETMASK=255.255.0.0 \
MAC=00:50:56:13:13:1a \
IP=172.20.6.25 \
DIST=QEMU-coldfire-test \
DISTREL=0.9.1-0.1 \
OS=ucLinux \
OSREL=2.x \
MEMSIZE=128 \
HDDBOOTIMAGE\_INST\_SIZE=2G \
HDDBOOTIMAGE\_INST\_BLOCKCOUNT=8 \
INST\_KERNEL=vmlinux-2.6.21-uc0 \
ctys-createConfVM \
  -t QEMU \
  --label=tst325 \
  --auto-all
```

4. Either edit the sections or just append the following to the 'tst324.ctys' configuration file, refer to provided README:

```
KERNELIMAGE=vmlinux-2.6.21-uc0
INITRDIMAGE=;

\# -nographic
VGA\_DRIVER=" -nographic "
CPU=any
NIC=;
```

5. Start virtual machine:

```
ctys -t qemu \
  -a create=1:tst325,b:\$PWD,reuse,console:gterm \
  -d pf,1 \
  app1'(-d pf,1)'
```

10.19 UnbreakableLinux-5.4

ffs.

.

11 Installed Systems

OS	name	Inst-VM	Media
Android-2.2	*)	QEMU	ISO
CentOS-5.0	tstxxx	QEMU, KVM	PXE,ISO
CentOS-5.4	tst131	QEMU, KVM	PXE,ISO
CentOS-5.5	inst012	KVM	ISO
Debian-4.0r3-ARM	tst102	QEMU	ISO
Debian-4.0r3	tst130	QEMU	PXE,ISO
Debian-5.0.0	tst210	KVM	PXE,ISO,debootstrap
Debian-5.0.6	inst013	KVM	ISO
Fedora8	tst240	KVM	PXE
Fedora10	tst239	KVM	ISO
Fedora12	tst211	KVM	ISO
Fedora13	inst011	KVM	ISO
FreeBSD-7.1	tst238	KVM	ISO
FreeBSD-8.0	tst218	KVM	ISO
Mandriva-2010	tst212	KVM	ISO
MeeGo-1.0	*)	QEMU	ISO
NetBSD-4.7	*)	KVM	ISO
ScientifiLinux-5.4.1	tst213	KVM	PXE,ISO
OpenBSD-4.0	tst124	QEMU	ISO
OpenBSD-4.3	tst127	QEMU	ISO
OpenBSD-4.7	*)	KVM	ISO
OpenSuSE-10.2	tst153	QEMU	PXE,ISO
OpenSuSE-11.2	tst214	KVM	PXE,ISO
OpenSuSE-11.3	inst014	KVM	ISO
OpenSolaris 2009.6	tst241	KVM	ISO
QNX-6.4.0	tst323	KVM	ISO
Solaris 10	tst217	KVM	ISO
Ubuntu-8.04-D	tst236	QEMU,KVM	ISO
Ubuntu-9.10-D	tst215	KVM	ISO
Ubuntu-10.10-D	inst015	KVM	ISO
uClinux-arm9	tst324	QEMU	(QEMU)
uClinux-coldfire	tst325	QEMU	(QEMU)

Table 2: Overview of Installed-QEMU/KVM-VMs

In addition various test packages with miscellaneous CPU emulations of QEMU are available. Example templates for integration scripts are provided for ARM, Coldfire, MIPS, and PPC.

.

12 Configuration Files

12.1 Directory Structure

The expected default directory structure for the assembly of the runtime call is as depicted in the following figure. VMs could be placed anywhere within the filesystem an are detected by the **ENUMERATE** action with provided **BASE** parameter.

```
$HOME
+---.ctys
|   |
|   +qemu       QEMU specific configuration files for user specific
|               settings, else the installed are used as default.
|
|
```

Figure 84: UnifiedSessionsManager QEMU file structure

The VMs could be installed anywhere, as long the configuration file and the wrapper file have the same filename prefix and allocated together with the boot image within the same directory. The naming convention provides the following variants:

1. The directoryname is the conffilename prefix.
2. The image filename and the conf filename have the same prefix.
3. The label is the conffile prefix.

12.2 Configuration File

The Initialization of the framework comprize mainly the bootstrap of initial hooks for a specific framework version.

The configuration file is sourced into the wrapper file, thus allowing some actual runtime variables set coallocated - with the for now - additionally set ENUMERATE parameters. In case of required runtime parameters these parameter has to be literally identical to their ENUMERATE peers.

12.3 Ctys-Wrapper File

The actions to perform whitin a wrapper script comprize

1. Initialization of the framework The Initialization of the framework comprize mainly the bootstrap of initial hooks for a specific framework version.
2. Assembly of the static generic call parts The next steps evaluate the dynamic call parameters as choosen by the user for this specific execution thread. First of all the parameters are extracted from the CLI and the generic part of the actual execution string is assembled.
3. Assembly of the specific call and final execution

Altough the wrapper script could be varied as required, the basic structure should be kept for simplicity.

12.4 Syntax Elements

For additional information on generated files refer to the description of **ctys-createConfVM(1)**

13 SEE ALSO

ctys-CLI(1) , *ctys-createConfVM(1)* , *ctys-plugins(1)* , *ctys-QEMU(1)* , *ctys-uc-CLI(7)* , *ctys-uc-QEMU(7)* , *ctys-uc-X11(7)* , *ctys-vhost(1)*
, *ctys-VNC(1)* , *ctys-uc-VNC(7)* , *ctys-X11(1)*

For GuestOS Setups:
ctys-uc-Android(7) , *ctys-uc-CentOS(7)* , *ctys-uc-MeeGo(7)*

References

For QEMU:
QEMU.org: "QEMU"

QEMU-KVM:1 "QEMU-KVM"

QEMU - Offcial Guest-Support: "Guest-Support"

For KVM:
Linux-KVM.org: "KVM"

Linux-KVM.org: "Guest-Support"

For HOWTOs on **Serial-Console** refer to:
van Emery: "Linux Serial Console HOWTO"
David S. Lawyer / Greg Hawkins: "Serial-HOWTO"
David S. Lawyer: "Text-Terminal-HOWTO"

Networking:
VDE-Virtual Distributed Ethernet: http://sourceforge.net/projects/vde
VirtualSquare: http://virtualsquare.org
VirtualSquare: Basic Networking

14 AUTHOR

Maintenance: <acue_sf1@sourceforge.net>
Homepage: <http://www.UnifiedSessionsManager.org>
Sourceforge.net: <http://sourceforge.net/projects/ctys>
Berlios.de: <http://ctys.berlios.de>
Commercial: <http://www.i4p.com>

15 COPYRIGHT

- for documents see GFDL-1.3 with invariant sections for license conditions,

This document is part of the **DOC package**,

- for documents and contents from DOC package see
  'Creative-Common-Licence-3.0 - Attrib: Non-Commercial, Non-Deriv'
  with optional extensions for license conditions.

For additional information refer to enclosed Releasenotes and License files.