OpenXT™ Developer Guide High-assurance isolation & security for virtual environments

Publication date 12 June 2014 (build 1550)

Table of Contents

Chapter 1. About This Document

This document is intended for software developers building extensions to the OpenXT platform. The SDK detailed in this chapter is subject to change without notice.

Chapter 2. Manually Adding Extension Packs to a OpenXT Installation

An official means of adding extension packs to a OpenXT installation is not yet available. As a workaround for our early customers, this chapter contains instructions and a set of scripts to walk you through manually adding customizations to a OpenXT release repository (`packages.main`).

2.1. Getting Started

This chapter assumes that you have a relatively advanced level of Linux experience. Every attempt has been made to be as explicit and thorough in these instructions as possible, but some minor details have been omitted, as they are assumed to be general knowledge.

To follow these instructions, create a directory somewhere on your system to hold your work. In this chapter we will refer to this directory as `${WORK_DIR}`. Copy your OpenXT installation ISO to this directory. Create a subdirectory to mount the ISO on. We'll call this `mnt`. Mount the ISO using a loopback device:

```
sudo losetup /dev/loopX ./<XT_iso_name>.iso
sudo mount /dev/loopX ./mnt
```

Copy the `packages.main` directory from the mounted ISO to the `${WORK_DIR}`. Also create a new directory which we'll call `packages.main.custom` which is where we'll build a new repository that will have our customizations. Create a directory `bin` to hold the scripts described in this document and add it to your shell's path. There are a number of ways to accomplish this, and depending on your shell, they are often different. For bash, it suffices to append the path to your PATH environment variable:

```
export PATH=${PATH}:${WORK_DIR}/bin
```

Finally, create a directory named `crypto` to hold your crypto keys and certs. Your `${WORK_DIR}` should now look like this:

```
${WORK_DIR}/bin
${WORK_DIR}/crypto
${WORK_DIR}/packages.main
${WORK_DIR}/packages.main.custom
${WORK_DIR}/mnt
${WORK_DIR}/OpenXT-<ver>-installer.iso
```

2.2. Certificates

Certificate management is a complex topic that is outside the scope of this document. It is assumed that if you or your organization manages their own certificate authority, then they have the appropriate protections and processes in place to manage, sign and distribute certificates. For the purposes of this tutorial, we will use a simple 512-bit RSA key (`priv.key`) and a self signed certificate (`cacert.pem`):

```
openssl genrsa -out ${WORK_DIR}/crypto/priv.key
openssl req -new -x509 -key ${WORK_DIR}/crypto/priv.key -out ${WORK_DIR}/crypto/cacert.pem -days 1095
```

OpenXT recommends consulting the OpenSSL certificate documentation* for authoritative information.

2.3. Customizing root file systems

The first step to customizing a OpenXT root file system is knowing what customizations you'd like to make. This will be specific to your project, so this tutorial uses a simple and generic example of adding a new CA certificate to

*http://www.openssl.org/docs/

the repository of trusted certificates. This allows for a system with these customizations to be upgraded over the air with a repository signed with a certificate tied to this new CA. This certificate must be installed into the control domain (dom0).

We start by copying the root filesystem from the release repository to our new repository in `packages.main.custom`:

```
cp ${WORK_DIR}/packages.main/dom0-rootfs.i686.xc.ext3.gz ${WORK_DIR}/packages.main.custom
```

We uncompress the root filesystem and mount it on a temporary directory:

```
gunzip ${WORK_DIR}/packages.main.custom/dom0-rootfs.i686.xc.ext3.gz
mkdir ${WORK_DIR}/mnt
sudo mount -o loop ${WORK_DIR}/packages.main.custom/dom0-rootfs.i686.xc.ext3 ${WORK_DIR}/mnt
```

We then add our new certificate to the trusted certificates in OpenXT:

```
sudo cp ${WORK_DIR}/crypto/cacert.pem ${WORK_DIR}/mnt/usr/share/xenclient/repo-certs/prod
pushd ${WORK_DIR}/mnt/usr/share/xenclient/repo-certs/prod
sudo ln -s cacert.pem $(openssl x509 -noout -hash -in cacert.pem).0
popd
```

Finally we unmount and recompress the updated root file system:

```
sudo umount ${WORK_DIR}/mnt
gzip ${WORK_DIR}/packages.main.custom/dom0-rootfs.i686.xc.ext3
```

2.3.1. The OPKG Package Manager

File systems for the VMs in OpenXT use `opkg` for package management. OpenXT highly recommends packaging your modifications to OpenXT using `opkg` packages. When installing packages into an extracted root file system using the method above, be sure to instruct `opkg` to install packages to the directory to which you extracted the root file system.

Note

The `opkg` binary supplied in the extracted root file system is a 32-bit i686 binary and might not be compatible with your development system.

2.4. Signing a repository

There are three files that together constitute the repository signature. As part of signing the modified repository, we will generate these files, explain their significance and how they are related.

2.4.1. XC-PACKAGES

The `XC-PACKAGES` file is a description of the individual files that make up the repository. Its format is as follows:

```
<shortname> <filesize> <sha256sum> <format> <required> <filename>                <unpackdir>
dom0        106147834  hash        ext3gz   required   dom0-rootfs.i686.xc.ext3.gz /
...
```

You can consult the `XC-PACKAGES` in the source repository at `${WORK_DIR}/packages.main/XC-PACKAGES` for a reference. The only bits that will change in this file are the hashes on the archives that you chose to modify. To assist in generating a new `XC-PACKAGES` file for your custom repository, we have provided the `gen-packages.sh` script.

This script should be downloaded and copied to the `${WORK_DIR}/bin` directory. If you've followed this document closely and used the recommended file paths, you should simply be able to run the `gen-packages.sh` script from

the root ${WORK_DIR} directory. If you've used different file paths, you can specify the locations of important files and directories on the command line. See the usage message from the script below:

```
gen-packages.sh [-s packages.main] [-d packages.main.custom] [-o out-file]
```

In short, the gen-packages.sh script copies all missing files from the original repository to the new one, leaving the ones you've modified intact. It then re-hashes all of the files from the repository and generates the XC-PACKAGES for your custom repository. The only difference between the original and your custom XC-PACKAGES should be the hashes.

The gen-packages.sh script is included in the OpenXT SDK.

2.4.2. XC-REPOSITORY

The XC-PACKAGES file contains information only about the supplied files that constitute the repository. The repository itself also needs data to describe it with regard to release numbers, versions for upgrade, and so on. Basically it needs identifying information and a way to tie this to the packages.

The XC-REPOSITORY file does exactly this. The important bits with regard to modifying the repository is recalculating the hash in the hash field of the XC-REPOSITORY file. The hash is of the XC-REPOSITORY file which has changed per our last step. To automate the generation of an XC-REPOSITORY file for your custom repository, we have provided the gen-repository.sh script.

Copy this script to your path. ${WORK_DIR}/bin is a good place. If you've followed these instructions closely simply running the gen-repository.sh script from within ${WORK_DIR} will generate a new XC-REPOSITORY file in ${WORK_DIR}/packages.main.custom. If you've used different file paths, you can specify the locations of important files and directories on the command line. The usage message for this script can be displayed with the -h option and it is identical to the gen-repository.sh script shown above.

Note

All information aside from the hash is copied verbatim from the XC-REPOSITORY file from the source packages.main repository. Changing the values of the other metadata should be done with caution and only if you know what you're doing.

Note

The gen-repository.sh script is included on the OpenXT SDK ISO.

2.4.3. XC-SIGNATURE

Finally, once the XC-REPOSITORY file has been generated, we sign it with our private key and attach the corresponding certificate for verification. The steps to do so are automated in the gen-signature.sh script. Copy this file on to your shell's path as described above. When executed with no options, the script assumes a directory layout as recommended in this document. Alternatively you can supply the script with the necessary information on the command line. Again the usage message is available through the -h option.

Once this signature has been generated, you can verify it using the XC-SIGNATURE file and the certificate of the authority from which the signing key/certificate were generated. In this example, we've used a self-signed certificate so these are the same. Again, a script to automate verification is provided, the verify-signature.sh script, and this script has a usage message that can be viewed by passing it the -h option.

Note

The gen-signature.sh script is included on the OpenXT SDK ISO.

2.5. Distributing a custom repository

There are a number of mechanisms by which your custom repository can be installed/distributed. The simplest is the Over-the-air (OTA) Upgrade mechanism. We also provide instructions for installation over PXE. The method for PXE and the authoring of a custom installation CD-ROM are similar, but we omit instructions for creating a custom install CD, as they are beyond the scope of this document.

2.5.1. Over-the-air (OTA) upgrade

An existing OpenXT installation will reject any upgrade repository that isn't signed with a key that is part of the database of trusted keys on the platform. Currently this database contains only the OpenXT production key. To configure an existing OpenXT installation such that it will accept an upgrade repository signed with any other key, you must add the corresponding CA certificate to its database of trusted certs. To accomplish this, first copy the CA cert on to the OpenXT platform, into the `/tmp` directory which is not read-only. Then install it using the following commands:

```
cp cacert.pem /usr/share/xenclient/repo-certs/prod/
pushd /usr/share/xenclient/repo-certs/prod
ln -s cacert.pem $(openssl x509 -noout -hash -in cacert.pem).0
popd
```

This creates the link necessary for the `openssl` libraries to locate the certificate. Your OpenXT systems should then be able to upgrade to the repository signed with your key.

2.5.2. PXE installation

To distribute your custom XC repository over PXE is a bit more complicated than the OTA upgrade method. It is, however, extremely useful. Begin by setting up a PXE environment using the instructions supplied in Appendix : "*Installing OpenXT Over a Network Using PXE*" in *OpenXT™ Engine Administrator Guide*.

Once you've verified your PXE configuration works with a stock OpenXT repository, you can copy your custom repository from `${WORK_DIR}/packages.main.custom` to your FTP or web server. Installing this repository will fail since the installer's root file system does not have the CA certificate for the key that your custom repository was signed with. Populating the installer's root file system with your CA's certificate is very similar to adding your CA cert to the control domain root file system from our example above. The only difference is we're installing the certificate on to a different root filesystem.

Extract the installer root filesystem from the installation ISO to a temporary directory and install the CA cert as we've done previously:

```
mkdir ${WORK_DIR}/tmp
pushd ${WORK_DIR}/tmp
gunzip ../mnt/isolinux/rootfs.gz | sudo cpio -i
cp ../crypto/cacert.pem ./usr/share/xenclient/repo-certs/prod
pushd ./usr/share/xenclient/repo-certs/prod
ln -s cacert.pem $(openssl x509 -noout -hash -in cacert.pem).0
popd
popd
```

Then package the root filesystem back up in a `gzip` archive:

```
pushd ${WORK-DIR}/tmp
sudo find . -print | sudo cpio -o -H newc | gzip > ../rootfs.gz
popd
```

The installer root filesystem will now be available in your `${WORK_DIR}` with the CA cert installed. You can then copy this on to your TFTP directory in place of the default `rootfs.gz` supplied in the stock OpenXT `isolinux` directory.

Chapter 3. Service VMs

3.1. Creating a service VM

This section describes how to create a service VM based on a Debian Linux distribution.

Note

Please use the latest stable i386 32-bit Debian release. Debian is currently the most-tested operating system for OpenXT service VMs. Using other operating systems might limit the capabilities of your service VM.

To Create a Service VM:

1. Create a VM using UIVM for OpenXT. Do *not* enable encryption for the VM disk.
2. In the OpenXT UI, select **Details > Advanced** for the VM and ensure that **Stub Domain** is set as *Enabled*.
3. Install the latest stable i386 32-bit Debian release. The following settings are recommended:
   - use the graphical installation
   - select the *SSH server* option
   - install grub as the bootloader

   Reboot the VM at the end of the installation.
4. Once the new VM starts up, log in.
5. Locate the file `xctools-debian-repo.tar.gz` provided in the OpenXT release and copy it to the service VM filesystem.
6. Unpack the archive:

```
gunzip xctools-debian-repo.tar.gz
tar -xvf xctools-debian-repo.tar
```

7. Edit `/etc/apt/sources.list` and add an entry to point to the unpacked Debian xctools repo:

```
echo "deb file:/<filepath>/debian wheezy main" >> /etc/apt/sources.list
```


Note

wheezy is the current version codename at the time of writing, which will change when the future version are available.

8. Install the Debian xctools:

```
apt-get update
apt-get install linux-image-3.11.10.4 v4v-module libv4v-1.0-0
```

9. Configure TTY login on the virtual serial console:

```
echo "S:2345:respawn:/sbin/getty 38400 hvc0" >> /etc/inittab
```

When you have finished configuring your service VM and it is running, you will be able to login to it through the virtual serial console. This requires use of the `screen` program in the control domain. To determine which `pts` in the control domain corresponds to the console in your service VM, execute the following command:

```
xenstore-read /local/domain/$(xec-vm --name <service_vm_name> get domid)/console/tty
```

You may then use the `screen` command to login to your service VM through this device.

Note

Use of the **screen** program is limited to the SELinux system administrator role. For instructions for entering into this role see Section 1.5.5: “The Effect of SELinux on Administrative Commands” in *OpenXT™ Engine Administrator Guide*.

10. Set the **v4v** module to auto-load:

```
echo "v4v" >> /etc/modules
```

11. Add the following lines to `/etc/init.d/ssh`:

- At the end of the *start* section, *before* the regular ssh daemon is invoked starts (**log_end_msg** exits the script).

```
LD_PRELOAD=/usr/lib/libv4v-1.0.so.0.0.0 INET_IS_V4V=1 /usr/sbin/sshd -p 2222
```

OpenXT advises against using the start-stop-daemon, which does not support this kind of environment variable declaration.

- At the end of the *stop* section, add:

```
killall sshd
```

12. Shut down the service VM.

13. In UIVM for OpenXT, press **Ctrl + Shift + T** to open a control domain terminal window.

14. Set the following firewall rules for the new VM:

```
xec-vm -n <service_vm_name> add-v4v-firewall-rule '0 -> myself:2222'
xec-vm -n <service_vm_name> add-v4v-firewall-rule 'myself:2222 -> 0'
```

15. Close the command prompt.

16. Back in UIVM for OpenXT, open the **Details** dialog for the service VM, select **Advanced**, and set the following properties:

- Under **Isolation Policies**:
  - **Stub Domain:** set to *Disabled*.
- Under **Virtual Compatibility**:
  - **Hardware Virtual Machine:** set to *Disabled*.
  - **Kernel Extraction Path:** set to *<disk_id_in_database>*,*<partition_number>*:/vmlinuz. (The default is *1,1:/vmlinuz*.)
  - **Command Line:** set to `root=/dev/xvda1 xencons=hvc0 console=hvc0 rw`.
  - **Initial Ramdisk:** leave this blank.

17. In UIVM for OpenXT, press **Ctrl + Shift + T** to open a control domain terminal window.

18. Enter the following commands to make the VM a service VM:

```
xec-vm -n <service_vm_name> --disk 1 set virt-path xvda
xec-vm -n <service_vm_name> set flask-label "system_u:system_r:nilfvm_t"
xec-vm -n <service_vm_name> set hvm false
xec-vm -n <service_vm_name> set qemu-dm-path ""
xec-vm -n <service_vm_name> set slot -1
xec-vm -n <service_vm_name> set type servicevm
```

Note

The flask-label assigned to the example service VM in this section is the label for the legacy NILFVM from the 2.x OpenXT releases. The permissions associated with this type are sufficient to boot the para-virtualized service VM and perform network interposition. We do however expect that developers will build service VMs for other purposes.

Custom service VMs will require custom policy suited to their function. A "one-size-fits-all" flask-label would contradict the isolation goals of the OpenXT flask policy. Currently developers requiring custom policy must modify and recompile the flask policy manually. The source code for the policy that ships with OpenXT is available on the provided source ISO. In the near future we expect to support a modular XSM policy in the same way we currently support a modular SELinux policy.

19. Boot the VM and verify that you can **sshv4v** to it:

```
xec-vm -n <service_vm_name> start
sshv4v <service_vm_name>
```


Note

If that doesn't work, grep *<service_vm_name>* in `/var/log/messages` to see if you can learn why.

3.2. Networking Backend Service VMs

This section introduces basic information about Service VMs that provide networking backends for guest VMs. This kind of Service VM can be used to process network packets flowing from guest VMs to OpenXT NDVMs. Typical uses are:

- packet filtering
- transparent VPN

There are no strict rules how network backend Service VMs should be built. The only thing that is required from a backend Service VM is that it can handle backend Virtual Interfaces (VIFs) of guest VMs and somehow provide network connectivity for these VIFs. This requires the `xen-netback` driver to be loaded (see the Procedure: "To Configure a Service VM to act as a NILF"). Typical network backend Service VMs create a bridge interface and attach all backend VIFs that will be created inside this VM to the bridge. This is typically done using a `udev` rule invoking a script that takes care of VIFs. See Figure 3.1: "VIF Plugging" for an example. The above "plumbing" is common for all the types of backend Service VMs. The next step is to provide network connectivity for the VMs attached to the bridge. This is simple in the transparent bridging case. Since transparent bridging works below layer 3 nothing needs to be done: it is enough to attach an interface provided by a network backend used by the backend Service VM to the bridge. In that case all packets, including DHCP configuration packets, will be transparently forwarded to a network backend for processing. The case of transparent bridging running over a VPN is similar. In the case of a VPN VM not using transparent bridging there are additional considerations:

- explicit routing rules or policy entries are required to direct packets through the VPN
- unless all the VMs using the network backend Service VM have static network configurations there is a need to run a DHCP server answering DHCP requests on the bridge interface
- depending on the VPN configuration used there may be a need to NAT packages exiting through the VPN
- VMs need to be given an address of a DNS server behind the VPN or the local DNS caching server running inside the network backend Service VM (in which case please ensure that the caching server uses a DNS server behind the VPN).

The next section provides a detailed guide about how to create a transparent packet filtering VM (NILF VM). OpenXT advises reading and understanding the entire section before attempting to create your own network backend Service VM.

3.3. Configuring NILF Functionality

A popular feature of the 2.X OpenXT release was the Network InLine Filter (NILF) VM. In replacing our custom NILF VM with a more generic toolkit for partners to use in creating their own Debian-based service VMs the NILF functionality was lost. This section provides instructions to recreate a NILF VM based on a Debian service VM as built with the instructions in Section 3.1: "Creating a service VM".

To Configure a Service VM to act as a NILF:

1. To function as a NILF, the Debian VM must first be configured to act as a network backend. This allows it to provide networking services to guests.

   Begin by adding the `xen-netback` driver to the modules list in your Service VM so that it is loaded on boot:

   ```
   echo "xen-netback" >> /etc/modules
   ```

2. Next install the packages required to create and configure a network bridge as well as those for interacting with xenstore:

   ```
   apt-get install bridge-utils xenstore-utils
   ```

3. With these utilities installed we can then add an `iface` block in `/etc/network/interfaces` to create a bridge when the VM boots. This is an example of a configuration block which creates a bridge and adds the `eth0` interface to it:

   ```
   auto brnilf
   iface brnilf inet manual
           up brctl addbr brnilf && brctl addif brnilf eth0 && ip link set up dev brnilf
           down brctl delif brnilf eth0 && ip link set down dev brnilf && brctl delbr brnilf
   ```

4. If you would like the NILF VM to be transparent on layer 3 you can modify the default network configuration for the `eth0` interface to prevent it from obtaining an address through dhcp:

   ```
   allow-hotplug eth0
   iface eth0 inet manual
           up ip link set up dev eth0
           down ip link set down dev eth0
   ```

5. Next we need a script to configure the network plumbing as virtual interfaces are added and removed from the NILF by the `xen-netback` driver. We provide an example script here which does two things:

   - The required communication with the toolstack through xenstore. This is signaling necessary to let the toolstack know that the network backend has been configured successfully.
   - Add the newly created `vif` to the bridge created in the previous step.

   This is only an example and it is expected that this script will be modified to meet the needs of your deployment.

Figure 3.1. VIF Plugging:

```
#!/bin/sh

TYPE=`echo "${XENBUS_PATH}" | cut -f 2 -d /`
DOMID=`echo "${XENBUS_PATH}" | cut -f 3 -d /`
DEVID=`echo "${XENBUS_PATH}" | cut -f 4 -d /`
XAPI="/xapi/${DOMID}/hotplug/${TYPE}/${DEVID}"
VIF="vif${DOMID}.${DEVID}"
BRIDGE=brnilf
NIC_OUT=eth0

logger -s "backend_vif_notify bridge for backend/${TYPE}/${DOMID}/${DEVID}: $1"

case "$1" in
add)
            mount -t xenfs xenfs /proc/xen 2>/dev/null

            xenstore-write "${XAPI}/vif" "${VIF}"
            xenstore-write "${XAPI}/hotplug" "online"

            if [ ! -d "/sys/class/net/${BRIDGE}" ] ; then
                brctl addbr "${BRIDGE}"
                ip link set up dev "${BRIDGE}"
                brctl addif "${BRIDGE}" "${NIC_OUT}"
            fi

            brctl addif "${BRIDGE}" "${VIF}"
            ip link set up dev "${VIF}"
            ;;

remove)
            ip link set down "${VIF}"
            brctl delif "${BRIDGE}" "${VIF}"

            xenstore-rm "${XAPI}/hotplug"
            ;;
esac
```

OpenXT recommends locating this script at `/etc/xen/scripts/vif-nilf`. Be sure to make this script executable:

```
chmod 755 /etc/xen/scripts/vif-nilf
```

6. We also need two udev rules to handle the creation of Xen devices:

```
SUBSYSTEM=="xen-backend", KERNEL=="vif*", RUN+="/etc/xen/scripts/vif-nilf $env{ACTION}"
KERNEL=="evtchn", NAME="xen/%k",SYMLINK+="xen/eventchn"
```

The first rule ties the `vif` creation events to the execution of the `vif-nilf` script. The second rule handles the creation of the Xen event channel device. Save these two udev rules to the file `/etc/udev/rules.d/xen-backend.rules`.

7. Finally guest VMs need to be configured to use this service VM as a network backend. This can be done by manually setting the backend UUID for the guest network.

   - Identify the UUID of the NILF VM:

     ```
     xec-vm --name <nilf_vm_name> get uuid
     ```

   - Set the `backend-uuid` for the guest to the UUID of the NILF VM:

     ```
     xec-vm --name <guest_vm_name> --nic 0 set backend-uuid <nilf-vm-uuid>
     ```

8. OpenXT recommends that the NILF VM be configured to boot when the platform

boots: `xec-vm --name <nilf_vm_name> set start-on-boot true`

And that the toolstack be configured to track dependencies for guests using the NILF:

```
xec-vm --name <guest_vm_name> set track-dependencies true
```


Note

It is important to note that the example `vif-nilf` script accounts for a situation where `vifs` are created in the NILF before the standard Debian `networking` scripts are able to create the bridge. The VM dependency tracking provided by OpenXT ensures that VMs are launched in the proper order, however it cannot guarantee all desired processing in the NILF will happen before the guest is started. `vif` scripts must be written to handle such situations with care.

3.4. Allowing Memory Introspection of Other VMs

Virtual Machine Introspection is a technique for externally monitoring the runtime state of a virtual machine. The runtime state can include CPU registers, memory, disk, network, and any other hardware-level events.

OpenXT can use a Service VM to perform memory introspection on any of the VMs running on it.

The following command allows a Service VM to perform memory introspection of other VMs on the host:

```
xec-vm -n <service_vm_name> set extra-xenvm xci-service=true
```

To disable this service:

```
xec-vm -n <service_vm_name> set extra-xenvm ""
```


Note

A service VM with which you intend to perform memory introspection must be built with the kernel compiled with the configuration option

```
CONFIG_STUBDOMAIN=y
```

Without this, the default privcmd driver in the service VM will prevent execution of the introspection hypercalls.

There are many ways to build a kernel; you can follow whatever procedure you are comfortable with. OpenXT recommends following the Debian documentation[*].

3.5. Setting Read-only Mode for a Disk on the Tapdisk Level

Although VMs can opt to use read-only filesystems just using their own logic, extra security can be achieved if read-only mode is toggled on the tapdisk level, so that it is actually guarded by the Control Domain.

To turn on read-only mode for a disk:

```
xec-vm -n <service_vm_name> --disk <disk_num> set mode r
```

To turn on read/write mode for a disk:

```
xec-vm -n <service_vm_name> --disk <disk_num> set mode w
```

This is enough for service VMs which do not use the template system. Some service VMs (for example, ndvm and uivm) are using the template system, and for them, modifying the config file directly is necessary. They are already read-only. If you want to enable read/write mode for development purposes, this can be achieved via editing the file

[*] http://www.debian.org/releases/stable/i386/ch08s06.html.en

`/usr/share/xenmgr-1.0/templates/kent/service-<vm_tag>` file, where *vm-tag* is *ndvm* or *uivm*. In that file, in the *json* section pertaining to the appropriate disk, *"mode": "r"* can be changed into *"mode": "w"*. To reset the default mode, edit the file again and change *"mode": "w"* back to *"mode": "r"*.

3.6. Adding a Service VM to Synchronizer XT

The example below demonstrates how the Debian service VM built in Section 3.1: “Creating a service VM” can be deployed via Synchronizer XT. This uses the command-line interface and assumes that the VHD file for the service VM has already been copied to the Synchronizer XT server.

To Deploy a Debian Service VM via Synchronizer XT:

1. First add the VHD file as a disk:

```
DISK_UUID=$(sync-admin add-disk -q <disk_name> <file_path>)
```

2. Write the VM configuration items to a file:

```
nic/0:network:/wired/0/bridged
v4v:0 -> myself,2222:true
v4v:myself -> 0,5556:true
v4v:myself,2222 -> 0:true
vm:apic:true
vm:cmd-line:root=/dev/hda1 xencons=xvc0 console=xvc0 rw
vm:boot:c
vm:flask-label:system_u:system_r:nilfvm_t
vm:hap:true
vm:hvm:false
vm:kernel-extract:0,1:/vmlinuz
vm:measured:false
vm:memory:256
vm:notify:dbus
vm:nx:true
vm:os:linux
vm:pae:true
vm:slot:-1
vm:stubdom:false
vm:type:servicevm
vm:v4v:true
vm:vcpus:1
vm:viridian:false
vm:xci-cpuid-signature:true
vmparam:acpi:true
```

3. Add a VM. The configuration items are listed in full here but could be read from a file instead:

```
VM_UUID=$(sync-admin add-vm -q -d "$DISK_UUID" -f <config_file> <service_vm_name>)
```

4. Finally assign the VM instance to a device:

```
sync-admin add-vm-instance <device_uuid> "$VM_UUID" <vm_instance_name>
```

Chapter 4. Windows VMs

4.1. Accessing Xen platform functionality from kernel mode

The XenPlatform package contains a library that can be used to access basic Xen platform functionality from kernel mode in Windows.

The package is included in the OpenXT SDK ISO. It consists of source code that can be built into a project to allow a custom driver to link with the Xen platform API:

File Name	Description
`xenplatform_api.h`	Definitions and comments for the Xen platform API functions.
`xenplatform_link.h`	Header defining the interface for accessing the Xen platform API.
`xenplatform_link.c`	Implementation of the interface for accessing the Xen platform API.
`xenplatform_samples.c`	Sample code showing how the Xen platform API is used.

Please refer to the header files for more in-depth information on the Xen platform API functions and the function used to link to the Xen platform API.

To use just one or two functions from the Xen platform API:

- Call `XenPlatformLink` to link to the Xen platform drivers and get the base pointer.
- Pass the base pointer to `XenPlatformResolveEntryPoint` to fetch a function pointer for as many of the functions defined in `XenPlatformFunctions` as are needed.
- When the Xen platform API is no longer needed, call `XenPlatformUnlink`.

Alternately, as a convenience, all the functions in the Xen platform API can be fetched in a single call to `XenPlatformRegisterApiFunctions`. This will return a struct `XenPlatformApiCalls` with the function pointers set. Note that if some functions are not found, they may be `NULL` in the structure, so this should be tested for.

Chapter 5. Database Access Control

You can access the control domain database using **db-*-dom0** commands in a service VM. This can be useful for creating persistent data, for example VPN certificates, in a secure location in the control domain. To be able to do database requests, you have to add the corresponding rules in `/etc/rpc-proxy.rules` on the control domain file system. For example to allow read access from a VM with the UUID `00000000-0000-0000-0000-000000000003`:

```
echo "allow dom-uuid 00000000-0000-0000-0000-000000000003 \
destination org.freedesktop.DBus interface \
org.freedesktop.DBus" >> /etc/rpc-proxy.rules

echo "allow dom-uuid 00000000-0000-0000-0000-000000000003 \
destination com.citrix.xenclient.db \
interface com.citrix.xenclient.db read" >> /etc/rpc-proxy.rules

echo "allow dom-uuid 00000000-0000-0000-0000-000000000003 \
destination com.citrix.xenclient.db \
interface com.citrix.xenclient.db read_binary" >> /etc/rpc-proxy.rules
```

To enable write access, replace `read` with `write` and `read_binary` with `write_binary` in the preceding example.

Chapter 6. VHD File Encryption

You can remove encryption from an encrypted VHD, or add encryption to an unencrypted VHD.

To Remove Encryption From a VHD:

Caution

Please do not include the `.vhd` suffix in the first command.

1. In UIVM for OpenXT, press **Ctrl + Shift + T** to open a control domain terminal window.
2. Enter the SElinux administrator role:

   ```
   newrole -r sysadm_r
   ```

3. Run the command:

   ```
   vhd-copy --src <source_vhd>.vhd --srckeydir <key_dir> --dest <destination_vhd>.vhd
   ```

   Where:

   - **source_vhd** is the name of the VHD file from which you wish to remove encryption.
   - **key_dir** is either `/config/platform-crypto-keys` if the VHD is from a local VM or `/config/sync/<synchroniser_name>` if the VHD is from a VM deployed by Synchronizer.
   - **destination_vhd** is the name of the unencrypted VHD to be created.

4. Delete the encrypted disk and key:

   ```
   rm <key_dir>/<source_vhd>,<encryption_type>,<key_size>.key
   rm <source_vhd>.vhd
   ```

5. Rename the plaintext VHD:

   ```
   mv <destination_vhd>.vhd <source_vhd>.vhd
   ```

To Encrypt a VHD:

1. In UIVM for OpenXT, press **Ctrl + Shift + T** to open a control domain terminal window.
2. Enter the SElinux administrator role:

   ```
   newrole -r sysadm_r
   ```

3. Run the commands:

   ```
   dd if=/dev/random of=<key_dir>/<source_vhd>,aes-xts-plain,512.key bs=1 count=64
   ```

   Caution

   Please do not include the `.vhd` suffix in the first command.

   Note

   The above command may take some time to return.

   ```
   vhd-copy --src <source_vhd>.vhd --dest <destination_vhd>.vhd --destkeydir <key_dir>
   ```

   Where:

   - **source_vhd** is the name of the VHD file to be encrypted.

- **key_dir** is either `/config/platform-crypto-keys` if the VHD is intended for a local VM or `/config/sync/<synchroniser_name>` if the VHD is intended for a VM deployed by Synchronizer.
- **destination_vhd** is the name of the encrypted VHD to be created.

4. Delete the unencrypted disk:

```
rm <source_vhd>
```

5. Rename the encrypted VHD:

```
mv <destination_vhd>.vhd <source_vhd>.vhd
```

Chapter 7. User Interface Branding Specification

The OpenXT goal is to provide anyone with the ability to incorporate their unique brand into the end user interface while still maintaining a unified "look and feel" across the range of virtual desktop solutions (whatever that means).

This chapter describes the various customizable components in the OpenXT user interface and provides guidelines and specifications describing how to properly format components for inclusion. All graphics within this document are included as examples only. Artwork files will be provided separately.

Included in OpenXT Branding Visual Assets zip folder are:

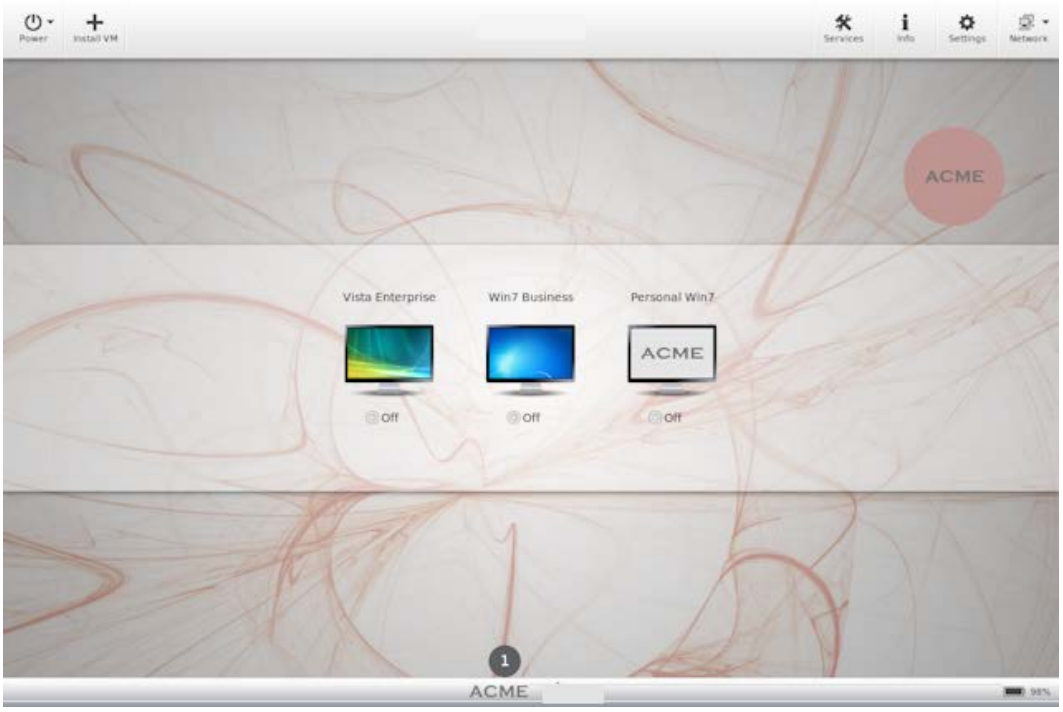
- A logo (for reference size and dimensions)
- Sample wallpaper (.png format)
- Virtual Machine Thumbnail (.png format)

7.1. Customizable components

1. **Company logo**
2. **Custom wallpaper**
3. **Wallpaper icon**

4. **Virtual machine thumbnail**
5. **Service logos**
6. **Custom message accessed via Info link**

7.1.1. Company logo specifications

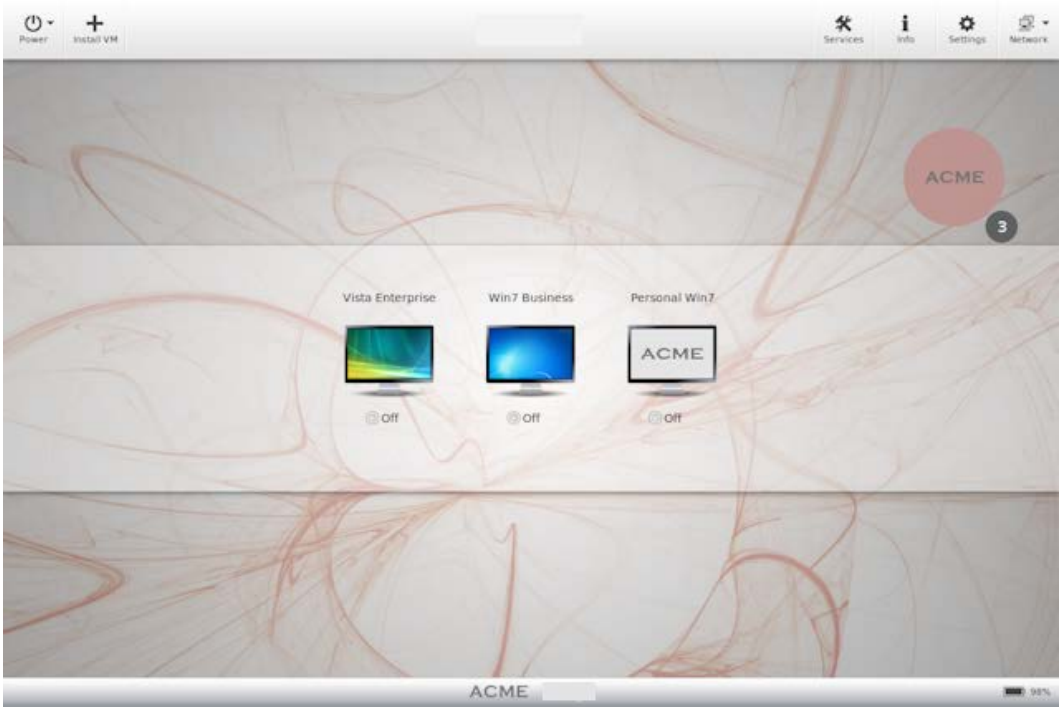


- Located at `plugins/branding/logo.png`
- Logo should be same height as the reference logo (25 pixels), maximum width of 200 pixels, and in png format
- Greyscale
- Transparent background with no border or box around it
- Placed in product screen

7.1.2. Custom wallpaper specifications

- Located in `plugins/wallpaper/`
- Dimensions: both 4:3 (1024 x 768 px) and 16:9 (1280 x 800 px) sizes are supported. Wallpaper thumbnail dimensions are 130 x 80 px.
- Color depth should not impact the visibility of the text shown around the virtual machine icons
- Any logo displayed as part of the wallpaper image should be no larger in any dimension than the virtual machine thumbnails
- 3 images need to be provided with each wallpaper (4:3, 16:9, thumbnail)

7.1.3. Wallpaper icon specifications

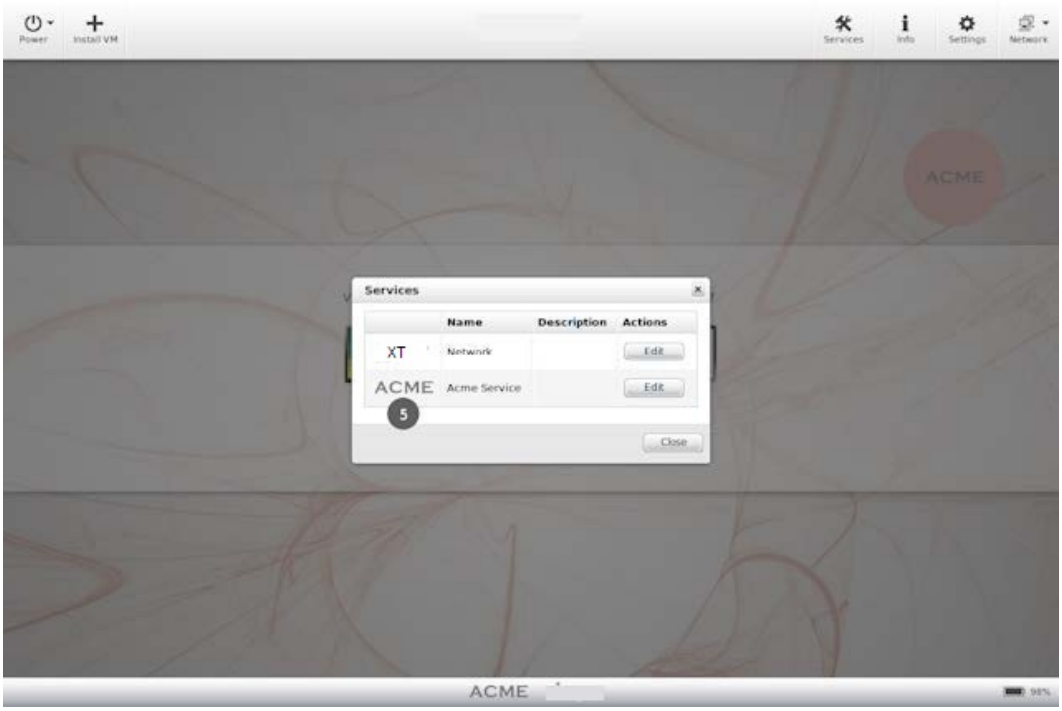


- Located in `plugins/branding/`
- The icon can be positioned in one of four positions. The desired position is indicated in the filename used:
  - Top right: `badge_TR.png`
  - Bottom right: `badge_BR.png`
  - Bottom left: `badge_BL.png`
  - Top left: `badge_TL.png`
- Color depth should not impact the visibility of the text shown around the virtual machine icons
- The logo in the wallpaper icon should be no larger in any dimension than the virtual machine thumbnails
- The wallpaper icon should use a transparent background to position the visible part the desired distance from the bounds of the user interface

7.1.4. Virtual Machine thumbnail specifications

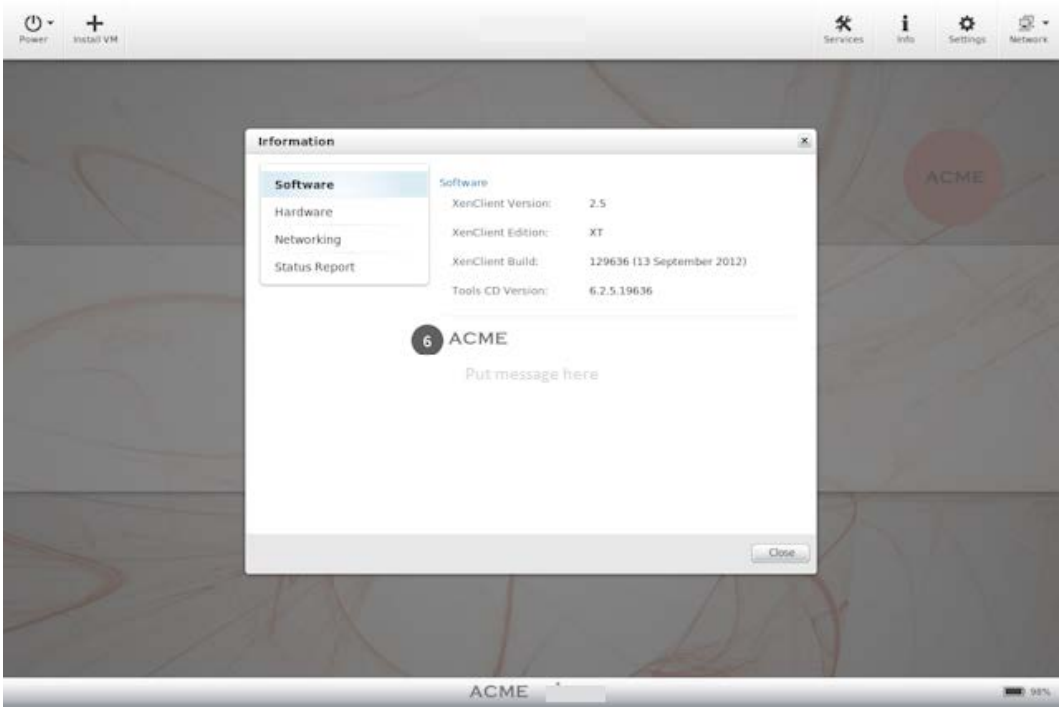
- Located in `plugins/vmimages/`
- Users can select a thumbnail from: Details > Icon
- The Virtual Machine thumbnail is 256 x 256 pixels, with the customizable area of the thumbnail (shown above as transparent) being 211 x 132 pixels. This thumbnail gets shown at smaller sizes throughout the user interface.
- Hovering over a VM thumbnail provides a magnifying animation, along with power control options to turn VM on/ off and a details link to the Advanced settings:

7.1.5. Service logo specifications



- Located in `plugins/serviceimages/`
- Logos should be same height as the logo (25 pixels), maximum width of 100 pixels, and in .png format
- Greyscale
- Transparent background with no border or box around it
- To display an image, the service virtual machine will need to have its image-path property set to *plugins/serviceimages/[filename].png*. This can be done using the CLI.

7.1.6. Custom messaging specifications



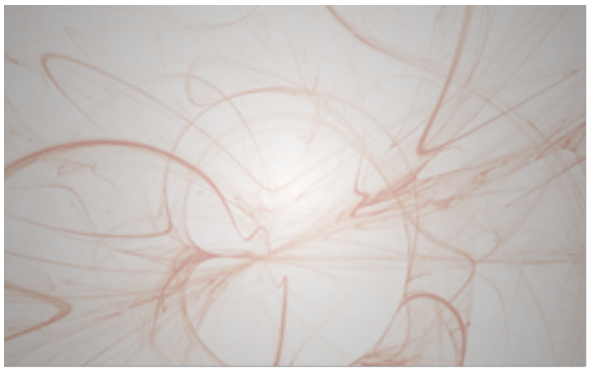
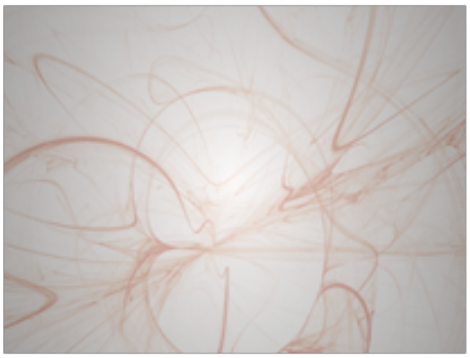
- Located in `plugins/branding/branding.html`
- Custom message can include basic HTML including lists and relative images - no hyperlinks or JavaScript can be included. Styles are outlined using an inline style block in the HTML. Refer to the example `branding.html` file.
- Custom messaging box size is 420 x 220 pixels. If the area grows larger than this, then it will scroll vertically.
- Custom messaging can be displayed from the System details layer, under Software
- Header text: Arial, 12 px, Bold, #333333
- Regular text: Arial, 12 px, #333333
- Bulleted lists may be used

7.2. Assets included in the sample zip file

All the sample assets used in the screenshots are included in the zip file. These include:

Logo (for reference)

Sample wallpaper (16:9, 4:3, thumbnail)


Sample wallpaper icon (top right)

Virtual Machine Thumbnail with blank screen (256 x 256 pixels)

Structure of the branding assets:

openxt.png

branding/

branding.html

logo.png

badge_TR.png

wallpaper/

acme_smoke_thumb.png

acme_smoke_1024.png

acme_smoke_1280.png

vmimages/

    acme_vm.png

    vm_image_template.png

serviceimages/

    acme_logo.png

Chapter 8. Input Plug-in API

The input plug-in API provides the ability to filter the input events, such as keyboard and mouse events, which would normally be sent to one VM and optionally redirect them to another VM.

This is based on a client/server architecture. The input server, a standard process which runs in the control domain, acts as the server. The input plug-in, a custom process which also runs in the control domain, acts as the client. The input server provides a Unix domain socket, `/var/run/input_socket`, to which the plug-in connects.

The plug-in can request to receive the input events for a particular VM. The input server will then send the plug-in any input events which it would normally send to the VM. In the plug-in does nothing with the events, no VM will receive them. However, the plug-in can send the events back to the input server, having first indicated which VM should receive them.

Note

OpenXT can be configured to display multiple VMs simultaneously on different monitors. In this configuration, when the user moves the mouse from one monitor to another, the VM on the first monitor will continue to receive keyboard events, but will no longer receive mouse events until the mouse is moved back to the first monitor. The VM on the second monitor will now receive mouse events, but will not receive keyboard events until the user clicks the mouse.

The same behavior occurs when the input plug-in is in use. If the plug-in has requested input events for the first VM, it will only receive keyboard events. If the plug-in has requested input events for the second VM, it will only receive mouse events.

8.1. Event Records

Input events are passed between the input server and the plug-in in the form of event records. The format is based on Linux EV event records, but has been extended with two additional event types. Each event record consists of the following fields:

Type	Name	Description
uint32	magic	This value is always `0xAD9CBCE9`. If reading the stream, you get a different value, it indicates a programmatic error.
uint16	type	Type of record, including EV_KEY (key/button), EV_REL (relative scalar), EV_ABS (absolute scalar), EV_SYN (Synchronization events.) or EV_DEV (Device change events).
uint16	code	This indicated which item (be it a key or axis) this record represents.
uint32	value	This is the new value for the item indicated by two field above. For a button or key, this would be a Boolean.

The event record format can accommodate a wide range of types of input event. Furthermore, events can be aggregated to form more complex events. Each input device is only able to send a small subset of all possible events. Events of type EV_DEV and EV_VM are not emitted by any device, but are an extension to the Linux EV system, to allow for the handling of multiple devices and multiple VMs.

The input server sends an EV_DEV event to the plug-in to indicate which device is emitting the events which follow, as well as to indicate which types of event are possible. For example, a tablet stylus emits absolute X and Y events, while a traditional mouse emits relative X and Y events.

The plug-in sends an EV_VM event as a command to the input server, either to request that it receives the events for a particular VM or to indicate which VM should receive any events it sends back to the input server. The input server may also send an EV_VM event to the plug-in as an error response to such a command.

The following sections describe the new events types EV_VM and EV_DEV and a subset of the standard event types. A complete list of event types can be found at here[*] with descriptions at here[†].

Details of the multitouch protocol can be found here[‡].

8.1.1. Type EV_VM (0x7): VM

This event type is an extension to the Linux EV system to support the input plug-in API.

EV_VM events represent commands sent from the plug-in to the input server and responses sent from the input server back to the plug-in.

Code	Name	Direction	Description
0x1	VM_SEND_TO	From plug-in	Sending this code to the input server indicates that all subsequent events should be sent to the specified VM. The VM is specified by domain ID.
0x2	VM_TAKE_FROM	From plug-in	Sending this code to the input server indicates that all subsequent events intended for the specified VM should be sent to to the plug-in. The VM is specified by domain ID. The plug-in can only ever take events from one VM, which will be the VM specified by the most recent use of this command. Sending a bad domain ID results in a "bad domain" error, and the plug-in not taking events from any domain.
0x3	VM_ERROR	From input server	The input server sends this to the plug-in to indicate that a previous command resulted in an error. The first byte is the error code (1 = bad domain, 2 = bad code). The second byte is the code from the failed command. The third byte is the domain ID from the failed command.

8.1.2. Type EV_DEV (0x6): Device Change

This event type is an extension to the Linux EV system to support the input plug-in API.

EV_DEV events describe actions relating to the device emitting the events which follow. EV_DEV events are not subject to synchronization events.

Code	Name	Description
0x1	DEV_SET	Events which follow originate from the device indicated by *value*. A *value* of -1 indicates that they are from an unknown source. Currently, keyboards always report -1, while other types of input device report a valid number.

[*] http://lxr.free-electrons.com/source/include/linux/input.h
[†] http://www.kernel.org/doc/Documentation/input/event-codes.txt
[‡] http://www.kernel.org/doc/Documentation/input/multi-touch-protocol.txt

Code	Name	Description
0x2	DEV_CONF	A new device has been created, which from this point will be identified as *value*. The number corresponds to the device node `/dev/input/eventN` in the control domain. This information is expected to be considered before attempting to decode the events for this device.
0x3	DEV_RESET	Device *value* is no longer valid, and any resources associated with this may be freed. If *value* is 0xFFFF, all devices are existing invalid — such an event is sent on start up.

8.1.3. Type EV_SYN (0x0): Synchronization

Code	Name	Description
0x0	SYN_REPORT	This is used as a barrier to indicate that all events of other types (excluding EV_VM and EV_DEV events) which occur between the same pair of SYN_REPORT events occurred at the same time. That is to say, events between pairs of SYN_REPORT events are to be aggregated. For example, when moving a mouse, it is common for both the X and Y coordinate to change together, so an event for each would be emitted, and because they were emitted between a pair of SYN_REPORT events, they would be aggregated. If however, only one axis changed, then only an event for that axis would be emitted. After this event has been added to the buffer, an interrupt is generated.
0x2	SYN_MT_REPORT	This is used for less able multitouch devices, to separate multiple sets of X/Y coordinates.
0x3	SYN_DROPPED	This informs the client when input events have been dropped from the EV_DEV input buffer due to a buffer overrun. The client should use this event as a hint to reset its state or ignore all following events until the next packet begins. The client should ignore all events up to and including next SYN_REPORT.

8.1.4. Type EV_REL (0x2): Relative

Code	Name	Description
0x0	REL_X	Movement on the X axis. **Note** Relative events are used before a mouse driver is installed in a VM.
0x1	REL_Y	Movement on the Y axis.  **Note** Relative events are used before a mouse driver is installed in a VM.

Code	Name	Description
0x8	REL_WHEEL	Vertical wheel movement.

8.1.5. Type EV_ABS (0x3): Absolute

Code	Name	Description
0x00	ABS_X	Position on the X axis.
0x01	ABS_Y	Position on the Y axis.
0x18	ABS_PRESSURE	The pressure with which the pen/tool is being applied.
0x2F	ABS_MT_SLOT	Each finger/pen in contact with the screen is given a slot number, and this is maintained until it is released. Multitouch events on the current device are henceforth for this slot. If only one finger/pen is in contact with the screen, then only one slot is required, and so this event may be omitted. If multiple fingers/pens are in contact with the screen, it would be common for many slot changes to occur during one pair of SYN_REPORT events, to indicate that each finger moved at the same time.
0x35	ABS_MT_POSITION_X	X coordinate for the current slot (finger/pen).
0x36	ABS_MT_POSITION_Y	Y coordinate for the current slot (finger/pen).
0x39	ABS_MT_TRACING_ID	Unique ID of initiated contact. A *value* of -1 indicates that the finger/pen has been released.

8.1.6. Type EV_KEY (0x1): Key/Button

Code	Name	Description
0x110	BTN_LEFT	The left button on a mouse.
0x111	BTN_RIGHT	The right button on a mouse.
0x112	BTN_MIDDLE	The middle button on a mouse.
0x113	BTN_SIDE	The side button on a mouse.
0x114	BTN_EXTRA	The extra button on a mouse.
0x115	BTN_FORWARD	The forward button on a mouse.
0x116	BTN_BACK	The back button on a mouse.
0x117	BTN_TASK	The task button on a mouse.
0x140	BTN_TOOL_PEN	The current tool is now a pen.
0x141	BTN_TOOL_RUBBER	The current tool is now a rubber.
0x145	BTN_TOOL_FINGER	The current tool is now a finger.

Code	Name	Description
0x146	BTN_TOOL_MOUSE	The current tool is now a mouse.
0x14A	BTN_TOUCH	The current tool touched the surface.
0x14B	BTN_STYLUS	The first button on the stylus was pressed.
0x14C	BTN_STYLUS2	The second button on the stylus was pressed.

8.2. Example

This section contains an example communication stream between the input server and the input plug-in. The plug-in first connects to the socket. It requests to receive events for the VM with domain ID 5 and indicates that any events it sends back should be directed to the VM with domain ID 6:

```
VM_TAKE_FROM 5
VM_SEND_TO 6
```

The input server sends the events which would otherwise have been sent to the VM with domain ID 5:

```
DEV_RESET 0xFFFF
DEV_CONF 4               # New device: associated config indicates it's a pen
DEV_CONF 7               # New device: associated config indicates it's a multi-touch device

DEV_SET 4                # Subsequent events from pen
ABS_X 345
ABS_Y 987
BTN_TOOL_PEN 1           # Pen hovering over the surface
SYN_REPORT

ABS_X 346
SYN_REPORT

ABS_Y 986
SNY_REPORT

ABS_X 344
ABS_Y 985
ABS_PRESSURE 45
BTN_TOUCH 1              # Pen touches the surface
SYN_REPORT

ABS_PRESSURE 48
SYN_REPORT

ABS_X 300
ABS_PRESSURE 20
SYN_REPORT

BTN_TOUCH 0              # Pen no longer touching the surface
ABS_X 388
SYN_REPORT

ABS_Y 810
ABS_X 320
BTN_TOOL_PEN 0           # Pen no longer hovering over the surface
SYN_REPORT

DEV_SET 7                # Subsequent events from multi-touch device
ABS_MT_SLOT 0            # First finger goes down
ABS_MT_TRACKING_ID 45
ABS_MT_POSITION_X 200
ABS_MT_POSITION_Y 300
SYN_REPORT
```

```
ABS_MT_POSITION_X 210    # Still on slot 0
SYN_REPORT

ABS_MT_POSITION_X 220
ABS_MT_POSITION_Y 302
SYN_REPORT

ABS_MT_POSITION_X 225
ABS_MT SLOT 1            # Second finger goes down
ABS_MT_TRACKING_ID 46
ABS_MT_POSITION_X 700
ABS_MT_POSITION_Y 800
SYN_REPORT

ABS_MT_SLOT 0
ABS_MT_POSITION_X 226
ABS_MT_POSITION_Y 308
ABS_MT_SLOT 1
ABS_MT_POSITION_Y 810
SYN_REPORT

ABS_MT_POSITION_Y 815    # Still on slot 1
ABS_MT_POSITION_X 720
SYN_REPORT

ABS_MT_SLOT 0            # First finger lifted
ABS_MT_TRACKING_ID -1
ABS_MT_SLOT 1
ABS_MT_POSITION_X 725
SYN_REPORT

ABS_MT_POSITION_Y 816    # Still on slot 1
ABS_MT_POSITION_X 740
SYN_REPORT

ABS_MT_TRACKING_ID -1    # Second finger is lifted
ABS_MT_POSITION_X 741
SYN_REPORT
```

As the plug-in reads this stream, it sends the events back to the input server, which directs them to the VM with domain ID 6. Each time the plug-in wishes to direct events to a different VM, it issues another VM_SEND_TO command.

Chapter 9. VM Event Hooks

OpenXT supports supplying custom RPC calls that run on the following events instead of the default product behaviour:

Event	Type
run-post-create	String.
run-pre-delete	String.
run-pre-boot	String.
run-insteadof-start	String.
run-on-state-change	String.
run-on-acpi-state-change	String.

To set the custom RPC calls for a VM, set the property on the VM via a comman-line call or using Synchronizer, for example:

```
xec-vm -n <vm_name> set run-insteadof-start \
"rpc:vm=<vm_uuid>,\
destination=com.citrix.xenclient.mepd,\
interface=com.citrix.xenclient.mepd,\
member=start_vm"
```

This example will cause a dbus call into the vm of UUID *<vm-uuid>* and given destination service, interface and member (method name) parameters, instead of starting the VM using the default product behaviour.

Chapter 10. OpenXT OVF (Open Virtualization Format) Support

OpenXT provides the **apptool** utility to support the installation, removal, and update of OVF appliances. An appliance is a set of virtual machine configurations along with other files such as virtual disk images, ISO images, encryption keys, and so on, packaged and managed as an unit. The **apptool** utility is installed on OpenXT by default.

apptool supports a subset of the upcoming 2.0 OVF specification, with OpenXT-specific extensions. The specification is available at here[*].

Note

All **apptool** commands must be run in the SELinux `sysadm_r` role. See Section 1.5.5: "The Effect of SELinux on Administrative Commands" in *OpenXT™ Engine Administrator Guide* for instructions about entering the `sysadm_r` role.

Note

In this section the use of the directory `/storage/import` is important for the proper functioning of the **apptool** command with SELinux. Be sure all files needed by your OVF are present in this directory. In order to get SELinux labels on imported files set properly, best practice is to download them onto the platform using the **wget** or **scp** command from within this directory. If any **apptool** commands fail producing SELinux AVC messages in `/var/log/messages` you should manually set the labels on the file tree rooted at `/storage/import` with the following command:

```
restorecon -r /storage/import
```

10.1. Basic apptool Usage

- To get some help information:

```
apptool --help
```

- To install an appliance from:
  - a directory containing OVF descriptor file and other files required by the appliance such as disks:

```
apptool --import /storage/import/appliance.ovf
```

  - OVA file (which is a tar gzipped directory containing OVF):

```
apptool --import /storage/import/appliance.ova
```

- To verify/scan the appliance for errors prior to importing (can also be done on any computer if the apptool is copied out of OpenXT):

```
apptool --verify /storage/import/appliance.ovf
```

- To list installed appliance IDs:

```
apptool --list
```

- To remove installed appliance:

```
apptool --remove <appliance_ID>
```

- To override appliance ID and version number while importing:

```
apptool --import /storage/import/appliance.ova --set-id=example-appliance --set-version=2
```

- To remove any partially-installed appliances in case of errors such as power interruption during installation:

[*] http://www.dmtf.org/standards/ovf

```
apptool --cleanup
```

Note

The OVF standard has no concept of appliance identifiers or versioning. The OpenXT implementation supports appliance ID and version using the OpenXT OVF extension (detailed later), or, in case of importing appliances not designed for OpenXT, overridden on the command line. Otherwise the appliance will be installed as `unnamed` with version defaulting to `1`. Installation of duplicate-ID appliances is prevented.

10.2. OVF 2.0 standard coverage

This section lists the parts of the OVF 2.0 standard which are covered by the OpenXT implementation and also points out those that are not.

10.2.1. OVF Package Structure Support

Feature	Supported?
Import of OVA packages or directories containing OVF file descriptors.	✓
The manifest file `.mf`. Checksums are verified if present.	✓
Disk image files.	✓
Extra environment files to be placed on environment ISO.	✓
Extra resource files, such as ISO images.	✓
Certificate files and manifest signature verification during import.	✗

10.2.2. Supported Virtual Disk Formats

The OVF specification does not require any specific disk format to be used. The format is given in form of URI string in the ovf package. OpenXT supports the following disk URI format strings:

Feature	Supported?
vhd - virtual hard disk format as used/extended by OpenXT	✓
cpiobz2 - bzipped2 CPIO file archive	✓
rawfilesystem - raw filesystem image	✓

10.2.3. Distribution as a Set of Files

Feature	Supported?
The distribution of appliance files from a web server (to be automatically downloaded by the import tool).	✗

10.2.4. Envelope element

Feature	Supported?
Standard envelope structure	✓
Specification of language using optional `xml:lang` and `Strings` elements.	✗

10.2.5. File References

Feature	Supported?
File references which are checked for existence during import.	✓
Relative file paths , for example, *subdir/disk1.vhd*. Such paths cross-reference to the file inside the OVF package.	✓
Absolute file path scheme, for example, *file:///storage/isos/xc-tools.iso*. Such paths cross-reference to the file already on the host (allowing the appliance to reference, for example, the OpenXT Tools iso which comes preinstalled with OpenXT).	✓
HTTP and HTTPS URL schemes.	✗
The `ovf:compression` attribute.	✗
The `ovf:chunkSize` attribute and file chunking.	✗

10.2.6. Content Element

Feature	Supported?
`VirtualSystem` and `VirtualSystemCollections` with nesting, as detailed by the OVF specification.	✓
Hierarchical relationships between systems and collections are not preserved. The hierarchy is flattened to a set of VMs during import.	✗

10.2.7. Extensibility

The importer will not reject sections which have **ovf:required="true"**, even if it does not understand them.

10.2.8. Virtual Hardware Section and the CIM classes

Feature	Supported?
Virtual hardware sections.	
Virtual hardware descriptors based on the following CIM classes: • CIM_ResourceAllocationSettingData • CIM_EthernetPortAllocationSettingData • CIM_StorageAllocationSettingData	
The virtual hardware descriptor based on CIM class `CIM_VirtualSystemSettingData`. Use VM properties or `ResourceAllocationSettingData` instead where appropriate.	
`ovf:configuration` attributes.	
`ovf:bound` attributes.	

10.2.8.1. CIM_ResourceAllocationSettingData

OpenXT supports setting memory size and number of vCPUs using the *CIM_ResourceAllocationSettingData* class.

VirtualQuantity and *VirtualQuantityUnits* CIM settings are meaningful.

10.2.8.2. CIM_StorageAllocationSettingData

Can be used in conjunction with the *StorageItem* element. Meaningful settings from this CIM class are:

- *InstanceID*
- *ResourceType*
- *AllocationUnits*
- *HostResource*
- *Access*
- *AutomaticAllocation*

Resource types supported:

- 15 (cd drive)
- 16 (dvd drive)
- 17 (hard disk)

Specifying backing using the *HostResource* element, either via `ovf:/file/<id>` or `ovf:/disk/<id>`, is supported. `ovf:/file/<id>` backing is only meaningful for optical drives in conjunction with ISO images.

10.2.8.3. CIM_EthernetPortAllocationSettingData

Can be used in conjunction with the *EthernetPortItem* element. Meaningful settings from this CIM class are:

- *InstanceID*
- *ResourceType*
- *Connection*
- *Address*
- *AutomaticAllocation*

10.2.8.4. Core Metadata Sections in Version 2

Feature	Supported?
`DiskSection`. Supported except for listed exceptions.	
`DiskSection` - Specification of parent leaf via `ovf:parentRef`.	
`DiskSection` - Specification of capacity (for empty disks) using property `ovf:capacity="${disk.size}"`.	
`SharedDiskSection` - full support only for readonly disk images.	
`NetworkSection`. Supported except for listed exceptions.	
`NetworkSection` - `NetworkPortProfiles`.	
`ProductSection`. Supported except for listed exceptions.	
`ProductSection` - property categories.	
`ProductSection` - icon elements.	
`ProductSection` - type constraints using `ovf:qualifiers`.	
`EulaSection`.	
`InstallSection`.	

Feature	Supported?
`EnvironmentFilesSection`.	
`ResourceAllocationSection`. Please use per-VM *VirtualHardwareSections* instead.	
`BootSection`. Due to qemu limitations, please use per vm "boot" property instead.	
`AnnotationSection`.	
`StartupSection`.	
`DeploymentOptionSection`.	
`OperatingSystemSection`.	
`ScaleOutSection`.	
`PlacementSection`.	
`EncryptionSection`.	

10.2.9. Internationalization

Feature	Supported?
Localizable messages, resource bundles and internationalization.	

10.2.10. OVF Environment

Supported. The *iso* transport method is supported, which creates a virtual CD-ROM device with an ISO containing the product properties and environment files. The product properties are propagated to all service VMs, not just the ones that are part of the imported appliance. This is different from the OVF specification, where properties are propagated only to VMs that are part of the same appliance, but allows for easier inter-appliance interoperability.

10.3. XCI/XT extensions to OVF

Compared to vanilla OVF, OpenXT appliances may optionally specify the *xci:ApplianceSection* section element, which contains extra information required to configure OpenXT-specific VM behavior. This element and all

children thereof use the *http://www.citrix.com/xenclient/ovf/1* namespace. The XML schema detailing contents of this element can be found in the OpenXT control domain at `/usr/share/apptool-1.0/schema/xciovf.xsd`.

The following is a sample *xci:ApplianceSection* using some of the available extensions:

```
<xci:ApplianceSection xci:applianceId="uivm" xci:version="1">
        <ovf:Info>XenClient appliance section</ovf:Info>
 <xci:Disk xci:ovfId="uivm-gconf">
   <xci:ImportEncryptionKey xci:fileRef="uivm-gconf-key"/>
 </xci:Disk>

 <xci:Disk xci:ovfId="uivm-swap" xci:filesystem="swap">
 </xci:Disk>

 <xci:VirtualMachine xci:ovfId="uivm" xci:uuid="00000000-0000-0000-0000-000000000001">
   <xci:V4VFirewall>
     <xci:V4VRule>myself -> 0:80</xci:V4VRule>
     <xci:V4VRule>myself -> 0:8080</xci:V4VRule>
     <xci:V4VRule>myself -> 0:5555</xci:V4VRule>
     <xci:V4VRule>myself -> dom-type=ndvm:5555</xci:V4VRule>
     <xci:V4VRule>myself -> 0:2222</xci:V4VRule>
     <xci:V4VRule>0 -> myself:2222</xci:V4VRule>
   </xci:V4VFirewall>

   <xci:RpcFirewall>
     <xci:RpcRule>allow destination org.freedesktop.DBus interface org.freedesktop.DBus</xci:RpcRule>
     <xci:RpcRule>allow destination org.freedesktop.ConsoleKit interface
     org.freedesktop.ConsoleKit.Manager member GetSessionForUnixProcess</xci:RpcRule>
     <xci:RpcRule>allow destination org.freedesktop.Hal</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.xenmgr</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.input</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.usbdaemon</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.updatemgr</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.surfman interface com.citrix.xenclient.surfman
   member increase_brightness</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.surfman interface com.citrix.xenclient.surfman
   member decrease_brightness</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.surfman interface com.citrix.xenclient.surfman
   member display_image</xci:RpcRule>
     <xci:RpcRule>allow destination com.citrix.xenclient.networkdaemon</xci:RpcRule>
   </xci:RpcFirewall>

   <xci:PropertyOverride>
     <xci:Property xci:name="slot" xci:value="0"/>
     <xci:Property xci:name="hidden-in-switcher" xci:value="true"/>
     <xci:Property xci:name="start-on-boot" xci:value="true"/>
     <xci:Property xci:name="start-on-boot-priority" xci:value="9"/>
     <xci:Property xci:name="provides-graphics-fallback" xci:value="true"/>
     <xci:Property xci:name="shutdown-priority" xci:value="-5"/>
     <xci:Property xci:name="hidden-in-ui" xci:value="true"/>
     <xci:Property xci:name="policy-modify-vm-settings" xci:value="false"/>

     <xci:Property xci:name="hvm" xci:value="false"/>
     <xci:Property xci:name="kernel-extract" xci:value="/boot/vmlinuz"/>
     <xci:Property xci:name="cmd-line" xci:value="root=/dev/hda xencons=xvc0"/>
     <xci:Property xci:name="flask-label" xci:value="system_u:system_r:uivm_t"/>
     <xci:Property xci:name="qemu-dm-path" xci:value="/usr/sbin/svirt-interpose"/>
   </xci:PropertyOverride>

   <!-- remember to change to true after testing!! -->
   <xci:DBEntry xci:key="measured" xci:value="false"/>

 </xci:VirtualMachine>
    </xci:ApplianceSection>
```

10.3.1. xci:ApplianceSection element

This is the primary element whose contents (and children) are used for specification of OpenXT-specific configuration. Since all extra configuration is contained in this element, and yet it needs to reference virtual machines, disk configurations specified elsewhere in the OVF doc are referenced using `xci:ovfId` attributes.

The *xci:ApplianceSection* element can optionally specify *xci:applianceId* and *xci:version* attributes. Although optional, giving the appliance a unique ID is recommended. If no ID is specified it will be imported as *unnamed*, or you can specify an unique ID during the import process (using the **apptool** command line).

xci:Disk, *xci:Network*, and *xci:VirtualMachine*elements can be child elements.

10.3.2. xci:Disk element

This element can be used to attach extra encryption information to the disk image files defined via *ovf:Disk* elements. This can take a form of a request to import a given encryption key, or request to generate an encryption key during appliance import. The latter is currently only meaningful for empty disks, or when importing raw filesystem disk images or **cpio** archives.

The following is an example import of an encryption key:

```
<xci:Disk xci:ovfId="uivm-gconf">
 <xci:ImportEncryptionKey xci:fileRef="uivm-gconf-key"/>
</xci:Disk>
```

The following is an example request to generate an encryption key:

```
<xci:Disk xci:ovfId="uivm-extra-runtime-data">
 <xci:GenerateEncryptionKey xci:keySize="512"/>
</xci:Disk>
```

Key sizes of 512 and 256 bits are supported, which correspond to AES-256 and AES-128 encryption as the secret is shared.

The *xci:Disk* element also supports an optional *filesystem* attribute, which can be useful when trying to add swap space to a VM, or while importing disk images packaged as **cpio** archives.

- `xci:filesystem="swap"` makes it easy to request that the import tool create a swap disk during appliance import.
- `xci:filesystem="ext3"`, or `ext4`, or `ntfs` hints the import tool to format the created VHD using specified filesystem type. It is useful when creating empty VHDs (avoiding the need to do it from within a VM on first boot), or when importing **cpio** disk images (which are just a set of files without filesystem information).
- The `xci:filesystem` attribute is ignored when specified on VHD disk images, since these are expected to already have preformatted contents.

10.3.3. xci:Network element

This element is used to map an OVF logical network onto networks exposed on the OpenXT device. For example:

```
<xci:Network xci:name="VM network 1" xci:clientNetworkId="/wired/0/bridged"/>
```

The `xci:name` attribute needs to be the same as the network name defined in the *ovf:Network* element, and the *xci:clientNetworkId* attribute needs to match a network using its ID as exposed by the network daemon on the OpenXT device.

10.3.4. xci:VirtualMachine element

This element is used to add extra configuration information, such as RPC firewall rules, PCI pass through rules, extra VM properties, and so on, to the VM configuration.

- The *xci:ovfId* attribute is required and must reference the *ovf:VirtualSystem* element.
- *xci:templateId* is optional and can specify one of the `/usr/share/xenmgr-1.0/templates/kent` VM creation templates to serve as the basic configuration. If omitted, the *new-vm-empty* template is used.
- *xci:uuid* is optional and can specify a fixed UUID of the imported VM. Importing VMs with duplicate UUIDs will fail during the verification stage. If omitted, a new UUID will be allocated.

xci:PropertyOverride, *xci:V4VFirewall*, *xci:RpcFirewall*, *xci:PCIPassthrough*, *xci:NetworkAdapter*, *xci:StorageDevice*, *xci:DBEntry*, and *xci:DomStoreFile* elements can be specified as children.

10.3.5. xci:PropertyOverride element

xci:PropertyOverride elements can be specified as children of either *xci:VirtualMachine*, *xci:NetworkAdapter*, or *xci:StorageDevice*. These elements can be used to override arbitrary DBUS properties (as given in the OpenXT IDL) for virtual machines, virtual network adapters and virtual storage devices (disks and optical devices), respectively.

For example, on a network adapter, this element can be used to specify the *backend-name* or *backend-uuid* property, useful for cases when a forced network backend VM is needed.

```
<xci:PropertyOverride>
 <xci:Property xci:name="backend-name" xci:value="vpnvm-1"/>
</xci:PropertyOverride>
```

10.3.6. xci:V4VFirewall element

This element allows specifying a list of child *xci:V4VRule* elements, which detail the v4v firewall rules in the same format as that in VM config files. Refer to the following example:

```
<xci:V4VFirewall>
 <xci:V4VRule>myself -> 0:80</xci:V4VRule>
 <xci:V4VRule>myself -> 0:8080</xci:V4VRule>
 <xci:V4VRule>myself -> 0:5555</xci:V4VRule>
 <xci:V4VRule>myself -> dom-type=ndvm:5555</xci:V4VRule>
 <xci:V4VRule>myself -> 0:2222</xci:V4VRule>
 <xci:V4VRule>0 -> myself:2222</xci:V4VRule>
</xci:V4VFirewall>
```

10.3.7. xci:PCIPassthrough element

This element allows specifying a list of matching rules which will be evaluated at VM startup. All PCI devices matching the rules will be passed through to the VM. The child elements can be a sequence of:

- *xci:MatchBDF* with required attribute *xci:bdf* to allow specifying the device in the BDF notation.
- *xci:MatchID* with optional attributes *xci:class*, *xci:vendor*, and *xci:device* to allow specifying device using PCI vendor/device IDs and PCI class. Omitted attributes are considered to be always matching.

Refer to the following example:

```
<xci:PCIPassthrough>
 <xci:MatchID xci:class="0x403" xci:vendor="0x8086"/>
 <xci:MatchBDF xci:bdf="0000:00:1a.0"/>
</xci:PCIPassthrough>
```

10.3.8. xci:NetworkAdapter element

This element has the required attribute *xci:ovfInstanceId*, which refers to the *InstanceID* attribute of the adapter, as defined in the VM's virtual hardware section. The *xc:PropertyOverride* child element may be specified.

10.3.9. xci:StorageItem element

This element has the required attribute *xci:ovfInstanceId*, which refers to the *InstanceID* attribute of the storage item, as defined in VM's virtual hardware section. The *xc:PropertyOverride* child element may be specified.

10.3.10. xci:DBEntry element

This element allows you to insert a value into either a VM's DB tree or domstore (domstore is the part of VM DB tree which can be accessed from that VM).

For example, the `measured` value is exposed as a read-only DBUS property, hence setting it with the `PropertyOverride` mechanism is not possible. But it can still be set via manipulation of VM's DB tree as follows:

```
<xci:DBEntry xci:key="measured" xci:value="false" />
```

- The *xci:section* attribute can be optionally specified to define whether value is being set in VM DB tree (`vm`, which is the default) or the domstore (`vm-domstore`).
- The *xci:key* and *xci:value* attributes are required.

10.3.11. xci:DomStoreFile element

This element allows you to request the importation of the file into domstore, where it can be subsequently read using a database operation in the VM. Note that these files are imported into the VM's config space, which is of limited size and should be kept small.

The *xci:fileRef* attribute references the file ID in the OVF *File* element.

```
<xci:DomStoreFile xci:fileRef="some-file-id"/>
```

10.4. Example Appliance Import

The OpenXT SDK ISO contains a minimal example of an OVF descriptor file named `squeeze-hvm.ovf`. This creates a VM with a single disk, 512 MB of RAM and 1 vCPU. As the name suggests, it is intended to create a VM running Debian Squeeze, but it is not specific to this and could be adapted to a different operating system.

This example assumes that you have created a VHD file named `squeeze.vhd` which represents a partitioned disk containing an installation of Debian Squeeze.

To Test the Import:

1. Ensure that you are running in the SELinux `sysadm_r` role. See Section 1.5.5: “The Effect of SELinux on Administrative Commands” in *OpenXT™ Engine Administrator Guide* for instructions about entering the `sysadm_r` role.
2. Ensure that the directory `/storage/import` exists and has the correct SELinux labels:

   ```
   mkdir -p /storage/import
   restorecon -r /storage/import
   ```

3. Copy the OVF descriptor file `squeeze-hvm.ovf` into `/storage/import`.

4. Copy the VHD file `squeeze.vhd` into `/storage/import`.
5. Import the appliance:

```
cd /storage/import
apptool --import squeeze-hvm.ovf
```

6. Verify that the VM `squeeze-vm` has been created:

```
xec-vm
```

7. Verify that the appliance `squeeze` is installed:

```
apptool --list
```

8. Click on the VM icon in the UI to verify that it can be started.

Chapter 11. API

Note

The API detailed in this chapter is subject to change without notice. The following table defines the data types used in the API documentation:

type	symbol	description
INVALID	0 (NUL)	Not a valid type code, used to terminate signatures
BYTE	y	8-bit unsigned integer
BOOLEAN	b	Boolean value, 0 is FALSE and 1 is TRUE. Everything else is invalid.
INT16	n	16-bit signed integer
UINT16	q	16-bit unsigned integer
INT32	i	32-bit signed integer
UINT32	u	32-bit unsigned integer
INT64	x	64-bit signed integer
UINT64	t	64-bit unsigned integer
DOUBLE	d	IEEE 754 double
STRING	s	UTF-8 string (must be valid UTF-8). Must be nul terminated and contain no other nul bytes.
OBJECT_PATH	o	Name of an object instance
SIGNATURE	g	A type signature
ARRAY	a	Array
STRUCT	r()	Struct
VARIANT	v	Variant type (the type of the value is part of the value itself)
DICT_ENTRY	e{}	Entry in a dict or map (array of key-value pairs)
UNIX_FD	h	Unix file descriptor

11.1. Interface com.citrix.xenclient.api.host

11.1.1. Methods:

11.1.1.1. Method shutdown

Shutdown the host device.

This method has no arguments.

11.1.1.2. Method reboot

Reboot the host device.

This method has no arguments.

11.1.1.3. Method sleep

Send the host into s3 (sleep).

This method has no arguments.

11.1.1.4. Method hibernate

Send the host into s4 (hibernate).

This method has no arguments.

11.1.2. Properties:

Name	Type	Access	Description
total_mem	i	read	Total memory, in megabytes.
free_mem	i	read	Free memory, in megabytes.
total_storage	i	read	Total storage, in megabytes.
free_storage	i	read	Free storage, in megabytes.
cpu-count	i	read	Total number of host CPU cores.
model	s	read	Host device model string.
vendor	s	read	Device vendor name.
bios-revision	s	read	Current device BIOS version.
physical-cpu-model	s	read	CPU type.
physical-gpu-model	s	read	GPU type.

11.2. Interface com.citrix.xenclient.api.product

11.2.1. Properties:

Name	Type	Access	Description
version	s	read	
build	s	read	Build number, date and associated information.

Name	Type	Access	Description
tools_version	s	read	

11.3. Interface com.citrix.xenclient.api.vm

11.3.1. Methods:

11.3.1.1. Method find_vm_by_uuid

Returns the object path to the VM with the given UUID, or raises an error.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	out	

11.3.1.2. Method list_templates

List the templates available for creating new VMs.

The arguments are:

Type	Name	Direction	Description
as	templates	out	

11.3.1.3. Method create_vm

Create a new VM.

The arguments are:

Type	Name	Direction	Description
o	path	out	

11.3.1.4. Method get_name

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	name	out	

11.3.1.5. Method set_name

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	name	in	

11.3.1.6. Method get_description

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	description	out	

11.3.1.7. Method set_description

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	description	in	

11.3.1.8. Method get_icon

Read a byte array representing the VM icon image.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
ay	icon	out	

11.3.1.9. Method add_disk

Add a new disk to VM.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
i	size	in	Size in megabytes.
o	path	out	

11.3.1.10. Method set_cdrom

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	cdrom	in	

11.3.1.11. Method get_wired_network

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	wired_network	out	

11.3.1.12. Method set_wired_network

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
s	wired_network	in	

11.3.1.13. Method get_has_tools

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
b	has_tools	out	

11.3.1.14. Method get_autostart

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
b	autostart	out	

11.3.1.15. Method set_autostart

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	
b	autostart	in	

11.3.1.16. Method switch

Switch to a VM.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	

11.3.1.17. Method start

Start the VM.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	

11.3.1.18. Method reboot

Reboot the VM.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	

11.3.1.19. Method shutdown

Shutdown the VM.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	

11.3.1.20. Method hibernate

s4 the VM.

The arguments are:

Type	Name	Direction	Description
s	vm_uuid	in	

11.3.2. Properties:

Name	Type	Access	Description
list_vms	ao	read	List each VM present: A list of dicts with few critical properties filled for each VM, including VM state, domain ID (if running), uuid etc.

11.4. Interface com.citrix.xenclient.usbdaemon

Interface to the USB policy daemon.

11.4.1. Methods:

11.4.1.1. Method get_policy_domuuid

Query the USB device policy for a specified VM.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	UUID of VM to query policy for.
s	value	out	Policy for specified VM, as XML.

11.4.1.2. Method set_policy_domuuid

Set the USB device policy for a specified VM.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	UUID of VM to query policy for.
s	policy	in	Policy for specified VM, as XML.

11.4.1.3. Method new_vm

Tell daemon a new VM has been started.

The arguments are:

Type	Name	Direction	Description
i	dom_id	in	ID of newly started VM.

11.4.1.4. Method vm_stopped

Tell daemon a VM is stopping.

The arguments are:

Type	Name	Direction	Description
i	dom_id	in	ID of stopping VM.

11.4.1.5. Method list_devices

Enumerate the USB devices plugged into the system.

The arguments are:

Type	Name	Direction	Description
ai	devices	out	List of IDs of devices in the system.

11.4.1.6. Method get_device_info

Connection states are defined as follows:

- -1 Cannot find device
- 0 Device not in use by any VM
- 1 Assigned to another VM which is off
- 2 Assigned to another VM which is running
- 3 Blocked by policy for specified VM
- 4 In use by specified VM
- 5 In use by specified VM and flagged "sticky"
- 6 Flagged as "sticky" assigned to specified VM, but not currently in use
- 7 Special platform device, listed purely for information
- 8 HiD device in use by dom0
- 9 HiD device in use by dom0, but "sticky" assigned to an off VM
- 10 External CD drive in use by dom0
- 11 External CD drive in use by dom0, but "sticky" assigned to an off VM

The arguments are:

Type	Name	Direction	Description
i	dev_id	in	ID of device to query.
s	vm_uuid	in	UUID of VM to get information relative to.
s	name	out	Name of device.

Type	Name	Direction	Description
i	state	out	Connection state of device.
s	vm_assigned	out	UUID of VM device is assigned to (if any).
s	detail	out	Name detail, for mouse-over text.

11.4.1.7. Method assign_device

Assign a device to a VM.

The arguments are:

Type	Name	Direction	Description
i	dev_id	in	id of device to assign
s	vm_uuid	in	UUID of VM to assign device to.

11.4.1.8. Method unassign_device

Unassign a device from a VM.

The arguments are:

Type	Name	Direction	Description
i	dev_id	in	ID of device to unassign.

11.4.1.9. Method set_sticky

Set or clear the sticky assignment flag for a device.

The arguments are:

Type	Name	Direction	Description
i	dev_id	in	ID of device to alter sticky flag for.
i	sticky	in	Set (1), or clear (0) sticky flag.

11.4.1.10. Method name_device

Set the user visible name for a device.

The arguments are:

Type	Name	Direction	Description
i	dev_id	in	ID of device to set name for.
s	name	in	name to set for device

11.4.1.11. Method state

Dump daemon state, for debugging only.

The arguments are:

Type	Name	Direction	Description
s	state	out	Daemon state in human readable form.

11.4.2. Signals:

11.4.2.1. device_rejected

A USB device has been rejected due to a policy.

The arguments are:

Type	Name	Direction	Description
s	device_name	in	User-friendly name of device rejected
s	reason	in	Human-readable description of reason for rejection

11.4.2.2. devices_changed

Device list previously given out may be out of date, re-enumerate.

This signal has no arguments.

11.4.2.3. device_info_changed

Information previously given out regarding device may be out of date, re-query.

The arguments are:

Type	Name	Direction	Description
i	dev_id	in	ID of device needing re-query.

11.5. Interface org.freedesktop.DBus.Introspectable

11.5.1. Methods:

11.5.1.1. Method Introspect

The arguments are:

Type	Name	Direction	Description
s	data	out	

11.6. Interface org.freedesktop.DBus

11.6.1. Methods:

11.6.1.1. Method Hello

The arguments are:

Type	Name	Direction	Description
s	a1	out	

11.6.1.2. Method RequestName

The arguments are:

Type	Name	Direction	Description
s	a1	in	
u	a2	in	
u	a3	out	

11.6.1.3. Method ReleaseName

The arguments are:

Type	Name	Direction	Description
s	a1	in	
u	a2	out	

11.6.1.4. Method StartServiceByName

The arguments are:

Type	Name	Direction	Description
s	a1	in	
u	a2	in	
u	a3	out	

11.6.1.5. Method UpdateActivationEnvironment

The arguments are:

Type	Name	Direction	Description
a{ss}	a1	in	

11.6.1.6. Method NameHasOwner

The arguments are:

Type	Name	Direction	Description
s	a1	in	
b	a2	out	

11.6.1.7. Method ListNames

The arguments are:

Type	Name	Direction	Description
as	a1	out	

11.6.1.8. Method ListActivatableNames

The arguments are:

Type	Name	Direction	Description
as	a1	out	

11.6.1.9. Method AddMatch

The arguments are:

Type	Name	Direction	Description
s	a1	in	

11.6.1.10. Method RemoveMatch

The arguments are:

Type	Name	Direction	Description
s	a1	in	

11.6.1.11. Method GetNameOwner

The arguments are:

Type	Name	Direction	Description
s	a1	in	

Type	Name	Direction	Description
s	a2	out	

11.6.1.12. Method ListQueuedOwners

The arguments are:

Type	Name	Direction	Description
s	a1	in	
as	a2	out	

11.6.1.13. Method GetConnectionUnixUser

The arguments are:

Type	Name	Direction	Description
s	a1	in	
u	a2	out	

11.6.1.14. Method GetConnectionUnixProcessID

The arguments are:

Type	Name	Direction	Description
s	a1	in	
u	a2	out	

11.6.1.15. Method GetConnectionDOMID

The arguments are:

Type	Name	Direction	Description
s	sender	in	
i	domid	out	

11.6.1.16. Method GetAdtAuditSessionData

The arguments are:

Type	Name	Direction	Description
s	a1	in	

Type	Name	Direction	Description
ay	a2	out	

11.6.1.17. Method GetConnectionSELinuxSecurityContext

The arguments are:

Type	Name	Direction	Description
s	a1	in	
ay	a2	out	

11.6.1.18. Method ReloadConfig

This method has no arguments.

11.6.1.19. Method GetId

The arguments are:

Type	Name	Direction	Description
s	a1	out	

11.6.2. Signals:

11.6.2.1. NameOwnerChanged

The arguments are:

Type	Name	Direction	Description
s	a1	in	
s	a2	in	
s	a3	in	

11.6.2.2. NameLost

The arguments are:

Type	Name	Direction	Description
s	a1	in	

11.6.2.3. NameAcquired

The arguments are:

Type	Name	Direction	Description
s	a1	in	

11.7. Interface com.citrix.xenclient.db

11.7.1. Methods:

11.7.1.1. Method read

The arguments are:

Type	Name	Direction	Description
s	path	in	
s	value	out	

11.7.1.2. Method read_binary

The arguments are:

Type	Name	Direction	Description
s	path	in	
ay	value	out	

11.7.1.3. Method write

The arguments are:

Type	Name	Direction	Description
s	path	in	
s	value	in	

11.7.1.4. Method dump

The arguments are:

Type	Name	Direction	Description
s	path	in	
s	value	out	

11.7.1.5. Method inject

The arguments are:

Type	Name	Direction	Description
s	path	in	
s	value	in	

11.7.1.6. Method list

The arguments are:

Type	Name	Direction	Description
s	path	in	
as	value	out	

11.7.1.7. Method rm

The arguments are:

Type	Name	Direction	Description
s	path	in	

11.7.1.8. Method exists

The arguments are:

Type	Name	Direction	Description
s	path	in	
b	ex	out	

11.8. Interface com.citrix.xenclient.fusechat

11.8.1. Methods:

11.8.1.1. Method list_desktops

The arguments are:

Type	Name	Direction	Description
s	credentials	in	
aa{ss}	desktops	out	

11.8.1.2. Method get_launch_ref

The arguments are:

Type	Name	Direction	Description
s	credentials	in	
s	app	in	
s	ref	out	

11.8.2. Properties:

Name	Type	Access	Description
server-url	s	readwrite	
pnagent-path	s	readwrite	

11.9. Interface com.citrix.xenclient.guest

Interface optionally implemented by the guest VM agent.

11.9.1. Methods:

11.9.1.1. Method request_shutdown

Request a clean guest shutdown.

This method has no arguments.

11.9.1.2. Method request_sleep

Request the s3 state.

This method has no arguments.

11.9.1.3. Method request_hibernate

Request the s4 state.

This method has no arguments.

11.9.1.4. Method request_reboot

Request a clean guest reboot.

This method has no arguments.

11.9.2. Signals:

11.9.2.1. agent_started

Sent by the guest agent after it starts.

This signal has no arguments.

11.9.2.2. agent_uninstalled

Sent by the guest agent during the uninstall process.

This signal has no arguments.

11.9.2.3. xorg_running

Sent by the guest agent when the x server is detected.

This signal has no arguments.

11.10. Interface com.citrix.xenclient.input

11.10.1. Methods:

11.10.1.1. Method set_slot

The arguments are:

Type	Name	Direction	Description
i	domid	in	
i	slot	in	

11.10.1.2. Method auth_set_context

The arguments are:

Type	Name	Direction	Description
s	user	in	
s	title	in	

11.10.1.3. Method auth_set_context_flags

The arguments are:

Type	Name	Direction	Description
s	user	in	
s	title	in	
i	flags	in	

11.10.1.4. Method auth_begin

Start authentication process.

The arguments are:

Type	Name	Direction	Description
b	started	out	

11.10.1.5. Method auth_remote_login

The arguments are:

Type	Name	Direction	Description
s	username	in	
s	password	in	

11.10.1.6. Method auth_collect_password

This method has no arguments.

11.10.1.7. Method auth_title

The arguments are:

Type	Name	Direction	Description
s	title	out	

11.10.1.8. Method auth_get_context

The arguments are:

Type	Name	Direction	Description
s	user	out	
s	title	out	
i	flags	out	

11.10.1.9. Method auth_remote_status

The arguments are:

Type	Name	Direction	Description
b	auto_started	in	
i	status	in	
s	id	in	
s	username	in	
s	recovery_key_file	in	

Type	Name	Direction	Description
u	ctx_flags	in	

11.10.1.10. Method auth_get_status

The arguments are:

Type	Name	Direction	Description
b	clear	in	
s	status	out	
i	flags	out	

11.10.1.11. Method auth_clear_status

This method has no arguments.

11.10.1.12. Method auth_create_hash

The arguments are:

Type	Name	Direction	Description
s	fname	in	
s	password	in	

11.10.1.13. Method get_user_keydir

Get the user crypto keys directory, or empty string if not mounted.

The arguments are:

Type	Name	Direction	Description
s	user	in	
s	dir	out	

11.10.1.14. Method get_remote_user_hash

Get the remote user hash of the specified userid.

The arguments are:

Type	Name	Direction	Description
s	userid	in	
s	hashed_userid	out	

11.10.1.15. Method auth_rm_platform_user

Used to recover access to platform when user has forgotten local password.

The arguments are:

Type	Name	Direction	Description
b	success	out	
s	error_msg	out	

11.10.1.16. Method get_focus_domid

Get currently focussed domain ID.

The arguments are:

Type	Name	Direction	Description
i	domid	out	

11.10.1.17. Method get_idle_time

Get Idle Time of the Host.

The arguments are:

Type	Name	Direction	Description
i	idle_time	out	

11.10.1.18. Method get_last_input_time

Returns the number of seconds since the most recent input activity or from start of input server.

The arguments are:

Type	Name	Direction	Description
i	idle_time	out	

11.10.1.19. Method switch_focus

Move focus to different domain.

The arguments are:

Type	Name	Direction	Description
i	domid	in	

Type	Name	Direction	Description
b	force	in	
b	success	out	

11.10.1.20. Method get_platform_user

The arguments are:

Type	Name	Direction	Description
s	user	out	
i	flags	out	

11.10.1.21. Method get_auth_on_boot

The arguments are:

Type	Name	Direction	Description
b	auth	out	

11.10.1.22. Method set_auth_on_boot

The arguments are:

Type	Name	Direction	Description
b	auth	in	

11.10.1.23. Method touchpad_get

The arguments are:

Type	Name	Direction	Description
s	prop	in	
s	value	out	

11.10.1.24. Method touchpad_set

The arguments are:

Type	Name	Direction	Description
s	prop	in	
s	value	in	

11.10.1.25. Method get_mouse_speed

The arguments are:

Type	Name	Direction	Description
i	mouse_speed	out	

11.10.1.26. Method set_mouse_speed

The arguments are:

Type	Name	Direction	Description
i	mouse_speed	in	

11.10.1.27. Method lock_timeout_set

The arguments are:

Type	Name	Direction	Description
i	value	in	

11.10.1.28. Method lock_timeout_get

The arguments are:

Type	Name	Direction	Description
i	value	out	

11.10.1.29. Method lock

The arguments are:

Type	Name	Direction	Description
b	can_switch_out	in	

11.10.1.30. Method get_kb_layouts

The arguments are:

Type	Name	Direction	Description
as	layouts	out	

11.10.1.31. Method get_current_kb_layout

The arguments are:

Type	Name	Direction	Description
s	layout	out	

11.10.1.32. Method set_current_kb_layout

The arguments are:

Type	Name	Direction	Description
s	layout	in	

11.10.1.33. Method update_seamless_mouse_settings

The arguments are:

Type	Name	Direction	Description
s	dom_uuid	in	

11.10.1.34. Method get_lid_state

Returns lid state. A value of one indicates lid is open otherwise closed.

The arguments are:

Type	Name	Direction	Description
u	lid_ret	out	

11.10.1.35. Method divert_mouse_focus

Diverts the mouse focus to the provided VM, for mouse events occurring within the given source frame. Events outside of this frame continue go to the calling VM. Mouse events are scaled to fit within the destination frame. Any area outside of the destination frame will not be accessible with the mouse. Coordinates: 0 is always left/top most, 0x1FFF right/bottom most, regardless of screen resolution.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
u	sframe_x1	in	
u	sframe_y1	in	
u	sframe_x2	in	
u	sframe_y2	in	
u	dframe_x1	in	

Type	Name	Direction	Description
u	dframe_y1	in	
u	dframe_x2	in	
u	dframe_y2	in	

11.10.1.36. Method stop_mouse_divert

This method stops a mouse focus divert, which was created using divert_mouse_focus.

This method has no arguments.

11.10.1.37. Method set_divert_keyboard_filter

This sets the keyboard filter, used with divert_keyboard_focus. Key_filter is an array of null terminated key value sets. Each key value set consists of a list of key values for any modifer keys, followed by a final activation key.

The arguments are:

Type	Name	Direction	Description
au	key_filter	in	

11.10.1.38. Method divert_keyboard_focus

Diverts the keyboard focus to the provided VM, filtering provided key strokes, which continue to go to the original VM. Key_filter is a null terminated array of key value sets, each of which is itself null terminated.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

11.10.1.39. Method stop_keyboard_divert

This method stops a keyboard focus divert, which was created using divert_keyboard_focus.

This method has no arguments.

11.10.1.40. Method touch

Introduce an input event, into the given VM, to defer it from sleeping.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

11.10.1.41. Method focus_mode

This bitfield allows the keyboard/mouse focus behaviour to be tweeked. Bit 1 =KEYFOLLOWMOUSE, Bit 2 = CLICKHOLDFOCUS, Bit 3 = CLONEEVENTS. The default mode is 0.

The arguments are:

Type	Name	Direction	Description
i	mode	in	

11.10.2. Properties:

Name	Type	Access	Description
numlock-restore-on-switch	b	readwrite	Restore NumLock status on switch to VM.

11.10.3. Signals:

11.10.3.1. keyboard_focus_change

This allows the monitoring of the keyboard focus, by signaling the UUID of the VM gaining focus.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

11.10.3.2. focus_auth_field

Sent to notify UI to focus the text field in secure mode.

The arguments are:

Type	Name	Direction	Description
i	field_id	in	

11.10.3.3. sync_auth_username

Sent to set the username in the UI in secure mode.

The arguments are:

Type	Name	Direction	Description
s	username	in	

11.10.3.4. auth_status

Sent to change the status in the UI in secure mode.

The arguments are:

Type	Name	Direction	Description
s	status	in	
i	flags	in	

11.10.3.5. secure_mode

Sent to the UI to enter secure mode.

The arguments are:

Type	Name	Direction	Description
i	show	in	

11.10.3.6. auth_remote_start_login

Sent to the BED to kick off Synchronizer auth.

The arguments are:

Type	Name	Direction	Description
s	username	in	
u	ctx_flags	in	

11.10.3.7. auth_remote_start_recovery

Sent to the BED to get recovery key.

The arguments are:

Type	Name	Direction	Description
b	auto_started	in	
s	id	in	
s	username	in	
u	ctx_flags	in	

11.10.3.8. lid_state_changed

Signals when a laptop lid is opened or closed.

This signal has no arguments.

NDVM Status.

11.11. Interface com.citrix.xenclient.networkdaemon

11.11.1. Methods:

11.11.1.1. Method add_vif

Creates the corresponding VIF.

The arguments are:

Type	Name	Direction	Description
u	domid	in	
u	backend_domid	in	
s	mac	in	

11.11.1.2. Method move_to_network

Move VIF to network.

The arguments are:

Type	Name	Direction	Description
s	vif	in	
s	network	in	

11.11.1.3. Method ndvm_status

To inform NDVM status.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
i	domid	in	
u	status	in	

11.11.1.4. Method shutdown

Shuts down the network daemon.

This method has no arguments.

11.11.1.5. Method is_networking_active

Returns true if any of the slave networks are up and running.

The arguments are:

Type	Name	Direction	Description
b	is_nw_active	out	

11.11.1.6. Method list

Lists networks.

The arguments are:

Type	Name	Direction	Description
aa{ss}	networks	out	

11.11.1.7. Method list_backends

Lists network backend domains.

The arguments are:

Type	Name	Direction	Description
as	backends	out	

11.11.1.8. Method is_initialized

Returns true if all the slave network configuration is complete.

The arguments are:

Type	Name	Direction	Description
b	is_initialized	out	

11.11.1.9. Method get_network_backend

Returns network backend for a network.

The arguments are:

Type	Name	Direction	Description
s	network	in	
s	uuid	out	

11.11.1.10. Method create_network

Creates network using configuration.

The arguments are:

Type	Name	Direction	Description
s	network_type	in	
i	id	in	
s	config	in	
s	network	out	

11.12. Interface com.citrix.xenclient.networkdaemon.notify

11.12.1. Signals:

11.12.1.1. networkdaemon_up

Notifies that the daemon is up.

This signal has no arguments.

11.12.1.2. network_added

Notifies that a new network is added.

The arguments are:

Type	Name	Direction	Description
s	network	in	

11.12.1.3. network_removed

Notifies that a network is no longer available.

The arguments are:

Type	Name	Direction	Description
s	network	in	

11.12.1.4. network_state_changed

Notifies that a network is no longer available.

The arguments are:

Type	Name	Direction	Description
s	network	in	
s	nm_state	in	

Type	Name	Direction	Description
s	backend	in	

11.13. Interface com.citrix.xenclient.networkdomain

11.13.1. Methods:

11.13.1.1. Method list_networks

Lists networks.

The arguments are:

Type	Name	Direction	Description
as	networks	out	

11.13.1.2. Method popup_network_menu

Signal the NM applet to popup the network menu.

The arguments are:

Type	Name	Direction	Description
u	x_off	in	
u	y_off	in	

11.13.1.3. Method close_network_menu

Signal the NM applet to close the network menu.

This method has no arguments.

11.14. Interface com.citrix.xenclient.networkdomain.config

11.14.1. Properties:

Name	Type	Access	Description
uuid	s	read	
domid	u	read	
nm-state	u	read	
name	s	read	
is-networking-active	b	read	

11.15. Interface com.citrix.xenclient.networkdomain.notify

11.15.1. Signals:

11.15.1.1. backend_state_changed

Notifies when a backend starts or stops.

The arguments are:

Type	Name	Direction	Description
u	status	in	

11.16. Interface com.citrix.xenclient.networkinterface

11.16.1. Methods:

11.16.1.1. Method name

Returns the name of the inteface.

The arguments are:

Type	Name	Direction	Description
s	name	out	

11.16.1.2. Method is_wireless

Returns true if the interface is wireless.

The arguments are:

Type	Name	Direction	Description
b	is_wireless	out	

11.16.1.3. Method mac_address

Returns the MAC address of the interface.

The arguments are:

Type	Name	Direction	Description
s	mac	out	

11.16.1.4. Method list_bridges

Lists the bridges to connect to use this interface.

The arguments are:

Type	Name	Direction	Description
a{ss}	bridges	out	

11.17. Interface com.citrix.xenclient.networkinterface.notify

11.18. Interface com.citrix.xenclient.network.nm

11.18.1. Methods:

11.18.1.1. Method popup_network_menu

Signal the NM applet to popup the network menu.

The arguments are:

Type	Name	Direction	Description
u	x_off	in	
u	y_off	in	

11.18.1.2. Method close_network_menu

Signal the NM applet to close the network menu.

This method has no arguments.

11.18.1.3. Method popup_keyboard

Pop up the keyboard in the uivm.

This method has no arguments.

Network types.

11.19. Interface com.citrix.xenclient.networkslave

11.19.1. Methods:

11.19.1.1. Method create_internal_networks

Method to create internal networks.

The arguments are:

Type	Name	Direction	Description
u	network_number	in	

11.19.1.2. Method backend_vif_notify

Method to notify network slave of a new backend VIF.

The arguments are:

Type	Name	Direction	Description
s	vif	in	
u	domid	in	
u	devid	in	

11.19.1.3. Method network_iface_notify

Method to notify network slave of the status of a networking interface.

The arguments are:

Type	Name	Direction	Description
s	udev_action	in	
s	interface	in	

11.19.1.4. Method move_vif_to_network

Move VIF to network.

The arguments are:

Type	Name	Direction	Description
s	vif	in	
s	network	in	

11.19.1.5. Method refresh_vifs

See that all the backend VIFs are connected to the right networks.

This method has no arguments.

11.19.1.6. Method shutdown

Shuts down the service.

This method has no arguments.

11.19.1.7. Method start_nm

Start Network Manager.

This method has no arguments.

11.19.1.8. Method is_initialized

Returns true if the system is initialized.

The arguments are:

Type	Name	Direction	Description
b	is_initialized	out	

11.19.1.9. Method nw_connectivity

Returns true if there is network connectivity.

The arguments are:

Type	Name	Direction	Description
b	is_initialized	out	

11.19.1.10. Method list_networks

Lists networks.

The arguments are:

Type	Name	Direction	Description
as	networks	out	

11.19.1.11. Method list_vifs

Lists backend VIFs.

The arguments are:

Type	Name	Direction	Description
as	vifs	out	

11.19.1.12. Method get_icavm_network

Returns icavm network object.

The arguments are:

Type	Name	Direction	Description
s	icavm_network	out	

11.19.1.13. Method nm_state

Returns network manager state.

The arguments are:

Type	Name	Direction	Description
u	nm_state	out	

11.20. Interface com.citrix.xenclient.networkslave.notify

11.20.1. Signals:

11.20.1.1. networkslave_up

Notifies NWD that the slave is up.

This signal has no arguments.

11.20.1.2. network_added

Notifies NWD that a new network is added.

The arguments are:

Type	Name	Direction	Description
as	network	in	

11.20.1.3. network_removed

Notifies NWD that a network is no longer available.

The arguments are:

Type	Name	Direction	Description
as	network	in	

11.20.1.4. new_backend_vif

Notifies NWD that a new backend VIF is added.

The arguments are:

Type	Name	Direction	Description
as	vif_info	in	

Network types.

Connection types.

Access point details used in extra_info method.

11.21. Interface com.citrix.xenclient.network

11.21.1. Methods:

11.21.1.1. Method is_configured

Returns true if the network is setup.

The arguments are:

Type	Name	Direction	Description
b	is_configured	out	

11.21.1.2. Method configure

Configure a shared/bridged network.

The arguments are:

Type	Name	Direction	Description
s	subnet	in	

11.21.1.3. Method join

Join the VIF to this network.

The arguments are:

Type	Name	Direction	Description
s	vif	in	

11.21.1.4. Method leave

Remove the VIF from this network.

The arguments are:

Type	Name	Direction	Description
s	vif	in	

11.22. Interface com.citrix.xenclient.network.config

11.22.1. Properties:

Name	Type	Access	Description
name	s	read	

Name	Type	Access	Description
bridge	s	read	
backend-uuid	s	read	
active	b	read	
interface	s	read	
mac-address	s	readwrite	
driver	s	read	
type	s	read	
connection	s	read	
nm-state	u	read	
extra-info	a{ss}	read	
label	s	readwrite	
nat-prefix	s	readwrite	
nm-managed	b	read	

11.23. Interface com.citrix.xenclient.network.notify

11.23.1. Signals:

11.23.1.1. state_changed

Notifies daemon of network state changes.

The arguments are:

Type	Name	Direction	Description
u	state	in	

11.24. Interface org.freedesktop.DBus.Introspectable

11.24.1. Methods:

11.24.1.1. Method Introspect

The arguments are:

Type	Name	Direction	Description
s	data	out	

11.25. Interface org.freedesktop.Hal.Manager

11.25.1. Methods:

11.25.1.1. Method GetAllDevices

The arguments are:

Type	Name	Direction	Description
as	devices	out	

11.25.1.2. Method GetAllDevicesWithProperties

The arguments are:

Type	Name	Direction	Description
a(sa{sv}	devices_with_props	out	

11.25.1.3. Method DeviceExists

The arguments are:

Type	Name	Direction	Description
b	does_it_exist	out	
s	udi	in	

11.25.1.4. Method FindDeviceStringMatch

The arguments are:

Type	Name	Direction	Description
as	devices	out	
s	key	in	
s	value	in	

11.25.1.5. Method FindDeviceByCapability

The arguments are:

Type	Name	Direction	Description
as	devices	out	

Type	Name	Direction	Description
s	capability	in	

11.25.1.6. Method NewDevice

The arguments are:

Type	Name	Direction	Description
s	temporary_udi	out	

11.25.1.7. Method Remove

The arguments are:

Type	Name	Direction	Description
s	udi	in	

11.25.1.8. Method CommitToGdl

The arguments are:

Type	Name	Direction	Description
s	temporary_udi	in	
s	global_udi	in	

11.25.1.9. Method AcquireGlobalInterfaceLock

The arguments are:

Type	Name	Direction	Description
s	interface_name	in	
b	exclusive	in	

11.25.1.10. Method ReleaseGlobalInterfaceLock

The arguments are:

Type	Name	Direction	Description
s	interface_name	in	

11.25.1.11. Method SingletonAddonIsReady

The arguments are:

Type	Name	Direction	Description
s	command_line	in	

11.25.2. Signals:

11.25.2.1. DeviceAdded

The arguments are:

Type	Name	Direction	Description
s	udi	in	

11.25.2.2. DeviceRemoved

The arguments are:

Type	Name	Direction	Description
s	udi	in	

11.25.2.3. NewCapability

The arguments are:

Type	Name	Direction	Description
s	udi	in	
s	cap_name	in	

11.25.2.4. GlobalInterfaceLockAcquired

The arguments are:

Type	Name	Direction	Description
s	interface_name	in	
s	lock_holder	in	
i	num_locks	in	

11.25.2.5. GlobalInterfaceLockReleased

The arguments are:

Type	Name	Direction	Description
s	interface_name	in	

Type	Name	Direction	Description
s	lock_holder	in	
i	num_locks	in	

11.26. Interface org.freedesktop.DBus.Introspectable

11.26.1. Methods:

11.26.1.1. Method Introspect

The arguments are:

Type	Name	Direction	Description
s	data	out	

11.27. Interface org.freedesktop.DBus.Properties

11.27.1. Methods:

11.27.1.1. Method Get

The arguments are:

Type	Name	Direction	Description
s	interface	in	
s	propname	in	
v	value	out	

11.27.1.2. Method Set

The arguments are:

Type	Name	Direction	Description
s	interface	in	
s	propname	in	
v	value	in	

11.27.1.3. Method GetAll

The arguments are:

Type	Name	Direction	Description
s	interface	in	
a{sv}	props	out	

11.28. Interface org.freedesktop.NetworkManager

11.28.1. Methods:

11.28.1.1. Method state

The arguments are:

Type	Name	Direction	Description
u	state	out	

11.28.1.2. Method wake

This method has no arguments.

11.28.1.3. Method sleep

This method has no arguments.

11.28.1.4. Method Enable

The arguments are:

Type	Name	Direction	Description
b	enable	in	

11.28.1.5. Method Sleep

The arguments are:

Type	Name	Direction	Description
b	sleep	in	

11.28.1.6. Method DeactivateConnection

The arguments are:

Type	Name	Direction	Description
o	active_connection	in	

11.28.1.7. Method ActivateConnection

The arguments are:

Type	Name	Direction	Description
s	service_name	in	
o	connection	in	
o	device	in	
o	specific_object	in	
o	active_connection	out	

11.28.1.8. Method GetDevices

The arguments are:

Type	Name	Direction	Description
ao	devices	out	

11.28.2. Properties:

Name	Type	Access	Description
State	u	read	
ActiveConnections	ao	read	
WirelessHardwareEnabled	b	read	
WirelessEnabled	b	readwrite	
NetworkingEnabled	b	read	

11.28.3. Signals:

11.28.3.1. StateChange

The arguments are:

Type	Name	Direction	Description
u		in	

11.28.3.2. DeviceRemoved

The arguments are:

Type	Name	Direction	Description
o		in	

11.28.3.3. DeviceAdded

The arguments are:

Type	Name	Direction	Description
o		in	

11.28.3.4. PropertiesChanged

The arguments are:

Type	Name	Direction	Description
a{sv}		in	

11.28.3.5. StateChanged

The arguments are:

Type	Name	Direction	Description
u		in	

11.29. Interface org.freedesktop.UPower.Device

11.29.1. Methods:

11.29.1.1. Method Refresh

This method has no arguments.

11.29.1.2. Method GetHistory

The arguments are:

Type	Name	Direction	Description
s	type	in	
u	timespan	in	
u	resolution	in	
a(udu)	data	out	

11.29.1.3. Method GetStatistics

The arguments are:

Type	Name	Direction	Description
s	type	in	
a(dd)	data	out	

11.29.2. Properties:

Name	Type	Access	Description
NativePath	s	read	
Vendor	s	read	
Model	s	read	
Serial	s	read	
UpdateTime	t	read	
Type	u	read	
PowerSupply	b	read	
HasHistory	b	read	
HasStatistics	b	read	
Online	b	read	
Energy	d	read	
EnergyEmpty	d	read	
EnergyFull	d	read	
EnergyFullDesign	d	read	
EnergyRate	d	read	
Voltage	d	read	
Luminosity	d	read	
TimeToEmpty	x	read	
TimeToFull	x	read	
Percentage	d	read	
IsPresent	b	read	

Name	Type	Access	Description
State	u	read	
IsRechargeable	b	read	
Capacity	d	read	
Technology	u	read	
RecallNotice	b	read	
RecallVendor	s	read	
RecallUrl	s	read	

11.29.3. Signals:

11.29.3.1. Changed

This signal has no arguments.

11.30. Interface org.freedesktop.UPower

11.30.1. Methods:

11.30.1.1. Method EnumerateDevices

The arguments are:

Type	Name	Direction	Description
ao	devices	out	

11.30.1.2. Method AboutToSleep

The arguments are:

Type	Name	Direction	Description
s	action	in	

11.30.1.3. Method Suspend

This method has no arguments.

11.30.1.4. Method SuspendAllowed

The arguments are:

Type	Name	Direction	Description
b	allowed	out	

11.30.1.5. Method Hibernate

This method has no arguments.

11.30.1.6. Method HibernateAllowed

The arguments are:

Type	Name	Direction	Description
b	allowed	out	

11.30.2. Properties:

Name	Type	Access	Description
DaemonVersion	s	read	
CanSuspend	b	read	
CanHibernate	b	read	
OnBattery	b	read	
OnLowBattery	b	read	
LidIsClosed	b	read	
LidIsPresent	b	read	
LidForceSleep	b	read	
IsDocked	b	read	

11.30.3. Signals:

11.30.3.1. DeviceAdded

The arguments are:

Type	Name	Direction	Description
o	device	in	

11.30.3.2. DeviceRemoved

The arguments are:

Type	Name	Direction	Description
o	device	in	

11.30.3.3. DeviceChanged

The arguments are:

Type	Name	Direction	Description
o	device	in	

11.30.3.4. Changed

This signal has no arguments.

11.30.3.5. Sleeping

This signal has no arguments.

11.30.3.6. NotifySleep

The arguments are:

Type	Name	Direction	Description
s	action	out	

11.30.3.7. Resuming

This signal has no arguments.

11.30.3.8. NotifyResume

The arguments are:

Type	Name	Direction	Description
s	action	out	

11.31. Interface com.citrix.xenclient.rpc_proxy

11.31.1. Methods:

11.31.1.1. Method validate_call

The arguments are:

Type	Name	Direction	Description
i	domain	in	
s	destination	in	

Type	Name	Direction	Description
s	interface	in	
s	member	in	
b	r	out	

11.31.1.2. Method validate_recv_signal

The arguments are:

Type	Name	Direction	Description
i	domain	in	
s	interface	in	
s	member	in	
b	r	out	

11.31.1.3. Method validate_send_signal

The arguments are:

Type	Name	Direction	Description
i	domain	in	
s	interface	in	
s	member	in	
b	r	out	

11.31.1.4. Method list_rules

The arguments are:

Type	Name	Direction	Description
as	rules	out	

11.31.1.5. Method add_rule

The arguments are:

Type	Name	Direction	Description
s	rule	in	

11.31.1.6. Method delete_rule

The arguments are:

Type	Name	Direction	Description
s	rule	in	

11.32. Interface com.citrix.xenclient.storehouse.sr

Storage repository interface.

11.32.1. Methods:

11.32.1.1. Method get_total_size

Total size of the storage repository in megabytes.

The arguments are:

Type	Name	Direction	Description
t	size	out	

11.32.1.2. Method get_free_size

Free size of the storage repository in megabytes.

The arguments are:

Type	Name	Direction	Description
t	size	out	

11.32.1.3. Method create_vdi

Create a virtual disk in repository.

The arguments are:

Type	Name	Direction	Description
t	size	in	
o	vdi	out	

11.32.1.4. Method create_vdi_with

Create a virtual disk in repository.

The arguments are:

Type	Name	Direction	Description
s	node	in	
o	vdi	out	

11.32.1.5. Method list_vdis

List virtual disks in repository.

The arguments are:

Type	Name	Direction	Description
ao	vdis	out	

11.32.1.6. Method list_nodes

List nodes in repository. The strings returned should be considered as an opaque blob.

The arguments are:

Type	Name	Direction	Description
as	nodes	out	

11.32.1.7. Method stream_download

Open a new stream to put a node in storehouse's grasp.

The arguments are:

Type	Name	Direction	Description
s	socket	out	

11.32.1.8. Method stream_upload

Open a new stream to get a node out of storehouse's grasp.

The arguments are:

Type	Name	Direction	Description
s	socket	out	

11.32.1.9. Method stream_close

Close a stream.

The arguments are:

Type	Name	Direction	Description
s	socket	in	

11.32.1.10. Method nodeGetChildren

Return the node's children if any.

The arguments are:

Type	Name	Direction	Description
s	name	in	
as	childs	out	

11.32.1.11. Method nodeGetParent

Return the node's parent if it exists.

The arguments are:

Type	Name	Direction	Description
s	name	in	
s	parent	out	

11.32.1.12. Method nodeDelete

Delete a node part of a historical chain of a VDI. May result in coalescing.

The arguments are:

Type	Name	Direction	Description
s	name	in	

11.32.1.13. Method nodeAddKey

Add a key associated with the node.

The arguments are:

Type	Name	Direction	Description
s	name	in	
ay	key	in	

11.32.1.14. Method nodeGetKey

Get the key associated with the node.

The arguments are:

Type	Name	Direction	Description
s	name	in	
ay	key	out	

11.32.1.15. Method nodeDelKey

Delete the key associated with the node.

The arguments are:

Type	Name	Direction	Description
s	name	in	

11.32.2. Properties:

Name	Type	Access	Description
name	s	readwrite	Human readable label.

11.33. Interface com.citrix.xenclient.storehouse.node

Nodes interface.

11.33.1. Methods:

11.33.1.1. Method get_parent

Return the node's parent if it exists.

The arguments are:

Type	Name	Direction	Description
s	uuid	out	

11.33.1.2. Method delete

Delete a node part of a historical chain of a VDI. May result in coalescing.

The arguments are:

Type	Name	Direction	Description
s	uuid	out	

11.33.1.3. Method make_vdi

Create a new VDI out of a node and returns the new VDI object.

The arguments are:

Type	Name	Direction	Description
o	uuid	out	

11.33.2. Properties:

Name	Type	Access	Description
creation_time	t	read	Creation time of this node.
immutable	b	readwrite	Node immutability.

11.34. Interface com.citrix.xenclient.storehouse.vdi

Virtual disk interface.

11.34.1. Methods:

11.34.1.1. Method open

Opens this VDI, making it accessible through a block device.

The arguments are:

Type	Name	Direction	Description
s	blkdev	out	

11.34.1.2. Method close

Closes this VDI, making it inaccessible through a block device.

This method has no arguments.

11.34.1.3. Method fork

Forks this VDI, returning a new VDI.

The arguments are:

Type	Name	Direction	Description
s	uuid	out	

11.34.1.4. Method snapshot

Takes a snapshot of the contents of this VDI at the current moment in time.

This method has no arguments.

11.34.1.5. Method revert

Revert a VDI to an older node in the chain.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

11.34.1.6. Method check

Check that a VDI is valid. returns OK or error name.

The arguments are:

Type	Name	Direction	Description
s	status	out	

11.34.1.7. Method list_nodes

Returns a list of all the nodes that constitute this VDI, ordered from newest to oldest.

The arguments are:

Type	Name	Direction	Description
as	uuid	out	

11.34.1.8. Method delete

Deletes this VDI.

The arguments are:

Type	Name	Direction	Description
b	delNodes	in	

11.34.2. Properties:

Name	Type	Access	Description
type	s	read	The type of this VDI: either disk or CD-ROM.

Name	Type	Access	Description
phys-type	s	read	The type of physical resource that backs this VDI: VHD, file, etc.

11.35. Interface com.citrix.xenclient.storehouse

Primary storage manager interface.

11.35.1. Methods:

11.35.1.1. Method dummy_method

Dummy method for testing.

The arguments are:

Type	Name	Direction	Description
s	dummy	in	

11.35.2. Properties:

Name	Type	Access	Description
dummy-prop	s	read	Dummy prop for testing.

11.36. Interface com.citrix.xenclient.surfman

11.36.1. Methods:

11.36.1.1. Method notify_death

The arguments are:

Type	Name	Direction	Description
i	domid	in	
i	sstate	in	

11.36.1.2. Method create_vgpu

The arguments are:

Type	Name	Direction	Description
i	domid	in	

Type	Name	Direction	Description
i	bus	in	
i	device	in	
i	function	in	

11.36.1.3. Method update_passthrough_bar

The arguments are:

Type	Name	Direction	Description
i	domid	in	
i	bar	in	
t	phys	in	
u	base	in	
u	size	in	

11.36.1.4. Method set_framebuffer_pages

The arguments are:

Type	Name	Direction	Description
i	domid	in	
b	dirty_tracking	in	
t	guest_addr	in	
at	pages	in	

11.36.1.5. Method set_framebuffer_paramters

The arguments are:

Type	Name	Direction	Description
i	domid	in	
i	width	in	
i	height	in	
i	stride	in	
s	format	in	

11.36.1.6. Method set_pv_display

The arguments are:

Type	Name	Direction	Description
i	domid	in	
s	be_type	in	

11.36.1.7. Method get_stride_alignement

The arguments are:

Type	Name	Direction	Description
u	stride	out	

11.36.1.8. Method get_heads

The arguments are:

Type	Name	Direction	Description
as	heads	out	

11.36.1.9. Method get_head_resolutions

The arguments are:

Type	Name	Direction	Description
i	head	in	
as	resolutions	out	

11.36.1.10. Method set_head_resolution

The arguments are:

Type	Name	Direction	Description
i	head	in	
s	resolution	out	

11.36.1.11. Method get_visible

The arguments are:

Type	Name	Direction	Description
ai	domids	out	

11.36.1.12. Method set_visible

The arguments are:

Type	Name	Direction	Description
ai	domids	in	
i	timeout	in	
b	force	in	

11.36.1.13. Method dump_all_screens

The arguments are:

Type	Name	Direction	Description
s	directoryname	in	

11.36.1.14. Method increase_brightness

This method has no arguments.

11.36.1.15. Method decrease_brightness

This method has no arguments.

11.36.1.16. Method pre_s3

This method has no arguments.

11.36.1.17. Method post_s3

This method has no arguments.

11.36.1.18. Method vgpu_mode

The arguments are:

Type	Name	Direction	Description
i	max_vgpus	out	
s	name	out	
b	msi_translation	out	

Type	Name	Direction	Description
as	bdfs	out	

11.36.1.19. Method has_vgpu

The arguments are:

Type	Name	Direction	Description
i	domid	in	
b	enabled	out	

11.36.1.20. Method get_surfaces_caching

The arguments are:

Type	Name	Direction	Description
b	caching	out	

11.36.1.21. Method display_image

The arguments are:

Type	Name	Direction	Description
s	filename	in	

11.36.1.22. Method display_text

The arguments are:

Type	Name	Direction	Description
s	text	in	

11.36.2. Signals:

11.36.2.1. start_service

Sent when surfman has started.

This signal has no arguments.

11.36.2.2. visible_domain_changed

Sent when visible domain has changed.

The arguments are:

Type	Name	Direction	Description
i	domid	in	

11.37. Interface com.citrix.xenclient.transfermgr

Transfer manager interface.

11.37.1. Methods:

11.37.1.1. Method list_transfers

List each transfer in progress. Return objects implementing com.citrix.xenclient.transfer.

The arguments are:

Type	Name	Direction	Description
ao	paths	out	

11.37.1.2. Method start_transfer

Download (if download parameter true) otherwise upload filename from/to URL username, password, certificate if not empty specify credentials for the transfer. Returns dbus path of transfer object implementing com.citrix.xenclient.transfer.

The arguments are:

Type	Name	Direction	Description
b	download	in	
s	url	in	
s	filename	in	
s	username	in	
s	password	in	
s	certificate	in	
o	path	out	

11.38. Interface com.citrix.xenclient.transfer

Transfer interface. See transfermgr to create or list. Note there is deliberately no password property. Transfers are not guaranteed to persist across reboots.

11.38.1. Methods:

11.38.1.1. Method pause

Pause transfer. No effect if the transfer is complete or already paused.

This method has no arguments.

11.38.1.2. Method resume

Resume transfer. No effect if the transfer is complete or already running.

This method has no arguments.

11.38.1.3. Method cancel

Cancel transfer. No effect if the transfer is complete.

This method has no arguments.

11.38.1.4. Method release

Release all records about the transfer.

This method has no arguments.

11.38.2. Properties:

Name	Type	Access	Description
filename	s	read	The dom0 absolute path of the file to read or write. No side effects.
url	s	read	The URL to download or upload. No side effects.
download	b	read	True for downloads, false for uploads. No side effects.
username	s	read	Empty string or a username for server authentication. No side effects.
certificate	s	read	Emptry string or the contents of a certificate to use to for openssl cacert to verify server SSL certificate. No side effects.
size	x	read	The total size in bytes of the transfer including content yet to transfer, or -1 if not known. No side effects.
transferred	x	read	The number of bytes transferred. No side effects.
running	b	read	True if the transfer is running; false if it has completed or paused. No side effects.
finished	b	read	True if the transfer has finished; false if it is paused or still running. No side effects.
error	s	read	A string describing an error encountered during the transfer, or an empty string when there are no errors. No side effects.

11.38.3. Signals:

11.38.3.1. progress

The arguments are:

Type	Name	Direction	Description
x	transferred	in	Number of bytes transferred.

11.38.3.2. completion

The arguments are:

Type	Name	Direction	Description
s	error	in	Error message or empty string on success.

11.38.3.3. paused

Invoked if the transfer is paused.

This signal has no arguments.

11.38.3.4. resumed

Invoked if the transfer is resumed.

This signal has no arguments.

11.39. Interface com.citrix.xenclient.updatemgr

Primary update manager interface.

11.39.1. Methods:

11.39.1.1. Method check_update

Download update metadata and check update applicability.

The arguments are:

Type	Name	Direction	Description
s	url	in	
s	version	out	XC version on the server (human-readable).
s	release	out	XC version on the server (strict format).

Type	Name	Direction	Description
s	status	out	Update applicability. Possible values are: 1. can-upgrade 2. cannot-upgrade 3. up-to-date

11.39.1.2. Method check_update_latest

Treat URL as a repository with multiple updates. We expect to find a text file with one URL per line at the given URL. We will treat the lines in reverse order, starting with the latest, and do a check_update on each. We'll report back on the first update found that is either 'up-to-date' or 'can-upgrade'. If all are 'cannot-upgrade', we return 'cannot-upgrade'. Report back on latest applicable update, if any.

The arguments are:

Type	Name	Direction	Description
s	url	in	
s	version	out	Latest applicable (or up-to-date) XC version on the server (human readable), if avaible. Empty if no version applicable.
s	release	out	Latest applicable (or up-to-date) XC version on the server (strict format), if avaible. Empty if no version.
s	status	out	Consolidated update applicability. Possible values are: 1. can-upgrade 2. cannot-upgrade 3. up-to-date

11.39.1.3. Method download_update

Start the download of the pending update.

The arguments are:

Type	Name	Direction	Description
s	url	in	

11.39.1.4. Method download_update_latest

Start the download of the latest pending update. See documenation for 'check_update_latest' for details.

The arguments are:

Type	Name	Direction	Description
s	url	in	

11.39.1.5. Method apply_update_and_reboot

Start the process of applying the update (if update pending, otherwise error). After success, reboot.

This method has no arguments.

11.39.1.6. Method apply_update_and_shutdown

Start the process of applying the update (if update pending, otherwise error). After success, shutdown.

This method has no arguments.

11.39.1.7. Method cancel_update

Cancel the update in progress (if possible, otherwise error).

This method has no arguments.

11.39.2. Properties:

Name	Type	Access	Description
update-url	s	read	Update's URL.
update-applicable	s	read	Whether currently selected update is applicable or not. Possible values: 1. can-upgrade 2. cannot-upgrade 3. up-to-date
update-state	s	read	State of pending update. Possible states are: 1. empty string when no update pending/nothing being done 2. downloading-meta 3. downloaded-meta 4. downloading-files 5. downloaded-files 6. applying 7. failed 8. done
update-description	s	read	Description of the pending update.
update-download-percent	d	read	Download percentage complete.

Name	Type	Access	Description
update-download-speed	d	read	Download speed.
update-fail-reason	s	read	Reason of failure in case something went wrong, otherwise empty.

11.39.3. Signals:

11.39.3.1. update_state_change

Singal that the update state has changed.

The arguments are:

Type	Name	Direction	Description
s	state	in	

11.39.3.2. update_download_progress

The arguments are:

Type	Name	Direction	Description
d	percent_comple	in	
d	speed	in	

11.40. Interface com.citrix.xenclient.vmdisk

All disk properties.

11.40.1. Methods:

11.40.1.1. Method attach_vhd

Makes this point to VHD image.

The arguments are:

Type	Name	Direction	Description
s	vhd_path	in	

11.40.1.2. Method attach_phy

Makes this point to physical disk.

The arguments are:

Type	Name	Direction	Description
s	phy_path	in	

11.40.1.3. Method mount

Mount in given directory for easy hacking.

The arguments are:

Type	Name	Direction	Description
s	dirpath	in	
b	readonly	in	

11.40.1.4. Method umount

Unmount if mounted.

This method has no arguments.

11.40.1.5. Method delete

Detach disk from VM and possibly remove VHD file.

This method has no arguments.

11.40.1.6. Method generate_crypto_key_in

Generate VHD's encryption key in specified directory.

The arguments are:

Type	Name	Direction	Description
i	keysize	in	
s	dirpath	in	

11.40.1.7. Method generate_crypto_key

Generate VHD's encryption key in platform's key directory.

The arguments are:

Type	Name	Direction	Description
i	keysize	in	

11.40.2. Properties:

Name	Type	Access	Description
backend-uuid	s	readwrite	UUID of the backend VM.
backend-name	s	readwrite	Name of the backend VM. Used only if backend-uuid is not set.
phys-path	s	readwrite	Path to VHD or /dev/sda1.
phys-type	s	readwrite	Type: vhd, phy, or others.
virt-path	s	readwrite	Path the guest sees.
mode	s	readwrite	Mode.
devtype	s	readwrite	Either cdrom or disk.
snapshot	s	readwrite	Snapshot mode.
shared	b	readwrite	Shared between multiple VMs (circumvents VHD deletion).
managed-disktype	s	readwrite	Type of managed disk, if relevant.
enabled	b	readwrite	Disk enabled/disabled flag.
encryption-key-set	b	read	Whether this VHD uses encryption key.
virtual-size-mb	i	read	Virtual size in megabytes.
utilization-bytes	x	read	Physical utilisation in bytes.

11.41. Interface com.citrix.xenclient.vmnic

VM Network Interface.

11.41.1. Methods:

11.41.1.1. Method delete

Delete NIC.

This method has no arguments.

11.41.2. Properties:

Name	Type	Access	Description
backend-uuid	s	readwrite	UUID of domain holding the driver backend.
backend-name	s	readwrite	Name of domain holding the driver backend. Used only if backend-uuid is not set.

Name	Type	Access	Description
network	s	readwrite	Network identifier.
wireless-driver	b	readwrite	Use wireless driver for this interface.
mac	s	readwrite	Specify MAC address, or 'auto'.
mac-actual	s	read	MAC address.
enabled	b	readwrite	Interface enabled/disabled flag.

11.42. Interface com.citrix.xenclient.xcpmd

11.42.1. Methods:

11.42.1.1. Method get_ac_adapter_state

Returns AC adapter state. A value of 1 means AC adapter in use, else 0.

The arguments are:

Type	Name	Direction	Description
u	ac_ret	out	

11.42.1.2. Method get_current_battery_level

Returns current battery level. 0 when normal, 1 for warning, 2 for low and 3 for critical.

The arguments are:

Type	Name	Direction	Description
u	battery_level	out	

11.42.1.3. Method get_current_temperature

Returns current platform temperature.

The arguments are:

Type	Name	Direction	Description
u	cur_temp_ret	out	

11.42.1.4. Method get_critical_temperature

Returns current critical platform temperature.

The arguments are:

Type	Name	Direction	Description
u	crit_temp_ret	out	

11.42.1.5. Method get_bif

Returns battery information as string.

The arguments are:

Type	Name	Direction	Description
s	bif_ret	out	

11.42.1.6. Method get_bst

Returns battery status as string.

The arguments are:

Type	Name	Direction	Description
s	bst_ret	out	

11.42.1.7. Method indicate_input

Called to indicate KB and other input values of interest.

The arguments are:

Type	Name	Direction	Description
i	input_value	in	Input values (since enum not implemented): INPUT_SLEEP=1 INPUT_BRIGHTNESSUP=2 INPUT_BRIGHTNESSDOWN=3.

11.42.1.8. Method hotkey_switch

Called to switch hotkey mapping values.

The arguments are:

Type	Name	Direction	Description
b	reset	in	TRUE to reset to boot time value, FALSE to set to platform specific value

11.42.2. Signals:

11.42.2.1. ac_adapter_state_changed

Signals change in platform AC adapter status.

This signal has no arguments.

11.42.2.2. battery_status_changed

Signals change in platform battery status.

This signal has no arguments.

11.42.2.3. battery_info_changed

Signals change in platform battery information.

This signal has no arguments.

11.42.2.4. power_button_pressed

Signals when power button is pressed.

This signal has no arguments.

11.42.2.5. sleep_button_pressed

Signals when sleep button is pressed.

This signal has no arguments.

11.42.2.6. oem_event_triggered

Signals when OEM special buttons/hotkeys are pressed.

This signal has no arguments.

11.42.2.7. battery_level_notification

Signals when battery level changes from normal.

This signal has no arguments.

11.42.2.8. bcl_key_pressed

Signals when brightness control hotkeys are pressed.

This signal has no arguments.

11.43. Interface com.citrix.xenclient.xenmgr.powersettings

Let host object implement the power settings toggles.

11.43.1. Methods:

11.43.1.1. Method get_ac_lid_close_action

Returns the action to perform when the user closes the laptop lid when the device is attached to a power source.

The arguments are:

Type	Name	Direction	Description
s		out	

11.43.1.2. Method get_battery_lid_close_action

Returns the action to perform when the user closes the laptop lid when the device is running on battery power.

The arguments are:

Type	Name	Direction	Description
s		out	

11.43.1.3. Method set_ac_lid_close_action

Sets the action to perform when the user closes the laptop lid when the device is attached to a power source.

The arguments are:

Type	Name	Direction	Description
s	action	in	

11.43.1.4. Method set_battery_lid_close_action

Sets the action to perform when the user closes the laptop lid when the device is running on battery power.

The arguments are:

Type	Name	Direction	Description
s	action	in	

11.44. Interface com.citrix.xenclient.xenmgr.host

11.44.1. Methods:

11.44.1.1. Method list_isos

List the contents of the CD image (ISO) directory.

The arguments are:

Type	Name	Direction	Description
as		out	

11.44.1.2. Method list_pci_devices

List the PCI devices on the system. Returns a list of mappings, each including at least the device name, pciclass, vendor and ID.

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.3. Method list_gpu_devices

Like above, with some logic to filter GPU devices.

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.4. Method list_disk_devices

List the disk devices on system.

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.5. Method list_playback_devices

List audio playback devices on system

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.6. Method list_capture_devices

List audio capture devices on system

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.7. Method list_sound_cards

List sound cards on system

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.8. Method list_sound_card_controls

List controlable parameters of a sound card

The arguments are:

Type	Name	Direction	Description
s	card	in	
aa{ss}		out	

11.44.1.9. Method get_sound_card_control

Set sound card control parameter value

The arguments are:

Type	Name	Direction	Description
s	card	in	
s	control	in	
s	value	out	

11.44.1.10. Method set_sound_card_control

Set sound card control parameter value

The arguments are:

Type	Name	Direction	Description
s	card	in	
s	control	in	
s	value	in	

11.44.1.11. Method list_cd_devices

List cdrom devices on system

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.44.1.12. Method assign_cd_device

Assign CD device to given VM (or remove assignment if vm param empty)

The arguments are:

Type	Name	Direction	Description
s	devid	in	
b	sticky	in	
s	vm_uuid	in	

11.44.1.13. Method get_cd_device_assignment

Get CD device's assigned VM

The arguments are:

Type	Name	Direction	Description
s	devid	in	
b	sticky	out	
s	vm_uuid	out	

11.44.1.14. Method eject_cd_device

Physically eject media tray

The arguments are:

Type	Name	Direction	Description
s	devid	in	

11.44.1.15. Method list_ui_plugins

List UI plugins on system.

The arguments are:

Type	Name	Direction	Description
s	subdir	in	

Type	Name	Direction	Description
as	list	out	

11.44.1.16. Method is_service_running

Tests if a named dbus service is currently running.

The arguments are:

Type	Name	Direction	Description
s	service	in	The name of the service to check for.
b	running	out	Boolean value indicating whether the service is running.

11.44.1.17. Method configure_gpu_placement

Configures GPU placement for purposes of tracking VM switching when moving the mouse over the monitor edge. 0 = not set.

The arguments are:

Type	Name	Direction	Description
s	id	in	GPU id
i	slot	in	Slot number, from left to right, in ascending order.

11.44.1.18. Method get_gpu_placement

Get the GPU placement slot.

The arguments are:

Type	Name	Direction	Description
s	id	in	GPU id
i	slot	out	GPU placement slot or 0 if not set.

11.44.1.19. Method get_seconds_from_epoch

Get number of seconds from epoch.

The arguments are:

Type	Name	Direction	Description
i	seconds	out	

11.44.1.20. Method shutdown

Shutdown the host device.

This method has no arguments.

11.44.1.21. Method reboot

Reboot the host device.

This method has no arguments.

11.44.1.22. Method sleep

Send the host into s3 (sleep).

This method has no arguments.

11.44.1.23. Method hibernate

Send the host into s4 (hibernate).

This method has no arguments.

11.44.1.24. Method set_license

The arguments are:

Type	Name	Direction	Description
s	expiry_date	in	Expiry date in the format yyyy-mm-dd HH:MM.
s	device_uuid	in	
s	hash	in	

11.44.2. Properties:

Name	Type	Access	Description
state	s	read	
total-mem	i	read	Total memory, in megabytes.
free-mem	i	read	Free memory, in megabytes.
avail-mem	i	read	Available memory, in megabytes. Can be bigger than free memory, since includes balloonable amount.
total-storage	i	read	

Name	Type	Access	Description
free-storage	i	read	
system-amt-pt	b	read	
cpu-count	i	read	
laptop	b	read	
model	s	read	
vendor	s	read	
serial	s	read	
bios-revision	s	read	
amt-capable	b	read	
eth0-mac	s	read	
eth0-model	s	read	
wireless-mac	s	read	
wireless-model	s	read	
physical-cpu-model	s	read	
physical-gpu-model	s	read	
safe-graphics	b	read	
measured-boot-enabled	b	read	
measured-boot-successful	b	read	
is-licensed	b	read	
build-info	a{ss}	read	Build number, date and associated information.
ui-ready	b	readwrite	UI please set this when you are ready. State will be stored in xenstore.
playback-pcm	s	readwrite	PCM device used for audio playback
capture-pcm	s	readwrite	PCM device used for audio capture

11.44.3. Signals:

11.44.3.1. state_changed

Notify that the host state has changed (sleeping, hibernating etc).

The arguments are:

Type	Name	Direction	Description
s	state	in	

11.44.3.2. storage_space_low

Notify that the user is running out of storage space.

The arguments are:

Type	Name	Direction	Description
i	percent_free	in	The percentage of free disk space remaining.

11.44.3.3. license_changed

Notify that licensing evaluation has changed.

This signal has no arguments.

11.45. Interface com.citrix.xenclient.xenmgr.installer

Helpers methods for the OEM installer.

11.45.1. Methods:

11.45.1.1. Method get_eula

Contents of the EULA, or empty string.

The arguments are:

Type	Name	Direction	Description
s	eula	out	

11.45.1.2. Method get_installstate

Get progress of the installation bits.

The arguments are:

Type	Name	Direction	Description
a{ss}	state	out	

11.45.1.3. Method progress_installstate

Set progress of the installation bits.

The arguments are:

Type	Name	Direction	Description
s	action	in	

11.46. Interface com.citrix.xenclient.xenmgr.vm

VM interface. Property access goes through policy checks.

11.46.1. Methods:

11.46.1.1. Method get_db_key

Get the value of a VM db key.

The arguments are:

Type	Name	Direction	Description
s	key	in	
s	value	out	

11.46.1.2. Method set_db_key

Set the value of a VM db key.

The arguments are:

Type	Name	Direction	Description
s	key	in	
s	value	in	

11.46.1.3. Method get_domstore_key

Get the value of a VM domstore (domain accessible disk-based private storage) key.

The arguments are:

Type	Name	Direction	Description
s	key	in	
s	value	out	

11.46.1.4. Method set_domstore_key

Set the value of a VM domstore (domain accessible disk-based private storage) key.

The arguments are:

Type	Name	Direction	Description
s	key	in	
s	value	in	

11.46.1.5. Method add_disk

Add a new disk to VM.

The arguments are:

Type	Name	Direction	Description
o	path	out	

11.46.1.6. Method list_disks

List the disk objects attached to this VM.

The arguments are:

Type	Name	Direction	Description
ao		out	

11.46.1.7. Method add_nic

Add a new NIC to the VM.

The arguments are:

Type	Name	Direction	Description
o	path	out	

11.46.1.8. Method list_nics

List the NIC objects attached to the VM.

The arguments are:

Type	Name	Direction	Description
ao		out	

11.46.1.9. Method delete

Remove a VM.

This method has no arguments.

11.46.1.10. Method switch

Switch to a VM.

This method has no arguments.

11.46.1.11. Method read_icon

Read a byte array representing the VM icon image.

The arguments are:

Type	Name	Direction	Description
ay	bytes	out	

11.46.1.12. Method start

Start the VM.

This method has no arguments.

11.46.1.13. Method start_internal

Start the VM, for internal use (bypass start hook)

This method has no arguments.

11.46.1.14. Method reboot

Reboot the VM.

This method has no arguments.

11.46.1.15. Method shutdown

Shutdown the VM.

This method has no arguments.

11.46.1.16. Method destroy

Force shutdown the VM.

This method has no arguments.

11.46.1.17. Method sleep

s3 the VM.

This method has no arguments.

11.46.1.18. Method hibernate

s4 the VM.

This method has no arguments.

11.46.1.19. Method resume

Wake the VM from s3.

This method has no arguments.

11.46.1.20. Method pause

Pause VM execution

This method has no arguments.

11.46.1.21. Method unpause

Resume VM execution from paused state

This method has no arguments.

11.46.1.22. Method suspend_to_file

Suspend the VM to disk.

The arguments are:

Type	Name	Direction	Description
s	file	in	

11.46.1.23. Method resume_from_file

Resume the VM from a suspended disk image.

The arguments are:

Type	Name	Direction	Description
s	file	in	

11.46.1.24. Method create_child_service_vm

Create a subordinate Service VM.

The arguments are:

Type	Name	Direction	Description
s	template	in	
o	path	out	

11.46.1.25. Method list_v4v_firewall_rules

The arguments are:

Type	Name	Direction	Description
as	rules	out	

11.46.1.26. Method add_v4v_firewall_rule

The arguments are:

Type	Name	Direction	Description
s	rule	in	

11.46.1.27. Method delete_v4v_firewall_rule

The arguments are:

Type	Name	Direction	Description
s	rule	in	

11.46.1.28. Method add_net_firewall_rule

The arguments are:

Type	Name	Direction	Description
i	id	in	
s	direction	in	
s	remoteip	in	
s	extra	in	

11.46.1.29. Method list_net_firewall_rules

The arguments are:

Type	Name	Direction	Description
aa{ss}	rules	out	

11.46.1.30. Method delete_net_firewall_rule

The arguments are:

Type	Name	Direction	Description
i	id	in	

11.46.2. Properties:

Name	Type	Access	Description
state	s	read	
acpi-state	i	read	
domid	i	read	
type	s	readwrite	
name	s	readwrite	
description	s	readwrite	
uuid	s	read	
seamless-id	s	readwrite	Handle seamless uses to reference the VMs.
slot	i	readwrite	
pv-addons	b	readwrite	
pv-addons-version	s	readwrite	
start-on-boot	b	readwrite	
start-from-suspend-image	s	readwrite	Start VM from suspend image file, if it exists.
time-offset	i	readwrite	
crypto-user	s	readwrite	
auto-s3-wake	b	readwrite	
os	s	readwrite	
image-path	s	readwrite	
wired-network	s	readwrite	
wireless-network	s	readwrite	
gpu	s	readwrite	
cd	s	readwrite	
mac	s	read	
amt-pt	b	readwrite	

Name	Type	Access	Description
portica-enabled	i	read	
portica-installed	b	read	
seamless-traffic	b	readwrite	
autostart-pending	b	read	
hibernated	b	read	
memory-static-max	i	readwrite	
memory-target	i	read	
memory-min	i	readwrite	
memory	i	readwrite	
hidden-in-switcher	b	readwrite	
hidden-in-ui	b	readwrite	
notify	s	readwrite	
hvm	b	readwrite	
pae	b	readwrite	
apic	b	readwrite	
viridian	b	readwrite	
hap	b	readwrite	
nx	b	readwrite	
cpuid	s	readwrite	Used to set/unset certain bits in response to CPUID instruction.
xci-cpuid-signature	b	readwrite	Advertise to guest as XenClient flavour of Xen.
sound	s	readwrite	
display	s	readwrite	
boot	s	readwrite	
cmd-line	s	readwrite	

Name	Type	Access	Description
kernel-extract	s	readwrite	[disk_number,partition_number:]/path/to/kernel Location of kernel image within the VM's disk(s). Examples: /boot/vmlinuz (/boot/vmlinuz in unpartitioned disk 0) 2:/boot/vmlinuz (/boot/vmlinuz in unpartitioned disk 2) 0,1:/boot/vmlinuz (/boot/vmlinuz in 1st partitin of disk 0)
kernel	s	readwrite	
initrd-extract	s	readwrite	[disk_number,partition_number:]/path/to/initrd Location of initrd image within the VM's disk(s). Examples: /boot/inird (/boot/inird in unpartitioned disk 0) 2:/boot/inird (/boot/inird in unpartitioned disk 2) 0,1:/boot/inird (/boot/inird in 1st partitin of disk 0)
initrd	s	readwrite	
acpi-pt	b	readwrite	Enable OEM windows install by exposing the ACPI SLIC table.
smbios-pt	b	readwrite	
vcpus	i	readwrite	
cores-per-socket	i	readwrite	
videoram	i	readwrite	
passthrough-mmio	s	readwrite	
passthrough-io	s	readwrite	
flask-label	s	readwrite	
qemu-dm-path	s	readwrite	
qemu-dm-timeout	i	readwrite	Timeout (in seconds) to wait for qemu response during qemu startup.
start-on-boot-priority	i	readwrite	Control priority of start-on-boot VMs, higher priority means earlier start. Defaults to 0 for new VMs.
shutdown-priority	i	readwrite	Controls the order of VM shutdown, higher priority means earlier shutdown. For new VMs defaults to 0, or -10 in case of PVM. VMs with the same priority are shutdown in parallel.
keep-alive	b	readwrite	Automatically restart this VM if it shuts down or crashes.

Name	Type	Access	Description
provides-network-backend	b	readwrite	Whether this domain is a networking backend and handles the physical NIC devices.
provides-default-network-backend	b	readwrite	Whether this domain is the primary networking backend and handles the physical NIC devices.
provides-graphics-fallback	b	readwrite	System will keep this VM on screen when no better candidate is present.
measured	b	read	If the VM is measured, the hash of the disk image is checked before it is allowed to start.
extra-xenvm	s	readwrite	Extra xenvm arguments, separated by semicolons.
extra-hvm	s	readwrite	Extra ioemu arguments, separated by semicolons.
crypto-key-dirs	s	readwrite	Comma-separated list of disk encryption keys directories.
dependencies	ao	read	VMs that are required to be running before starting this one.
track-dependencies	b	readwrite	If true, automatically start required VMs.
seamless-mouse-left	i	read	Where to move mouse when it goes over the left edge of a surface.
seamless-mouse-right	i	read	Where to move mouse when it goes over right edge of a surface.
control-platform-power-state	b	readwrite	If set, the entire platform will go into S3/S4/S5 when this VM goes into S3/S4/S5.
oem-acpi-features	b	readwrite	Enables access to additional OEM acpi features.
stubdom	b	readwrite	Use stub domain to hide the device emulator.
usb-enabled	b	readwrite	Enables PV USB support.
usb-control	b	readwrite	Enables the VM to communicate with the USB daemon.
usb-grab-devices	b	readwrite	Automatically assign all available USB devices upon this VM start.
greedy-pciback-bind	b	readwrite	Bind passthrough pci devices to pciback early, to prevent their use by other VMs.
policy-modify-vm-settings	b	readwrite	Allow modification of VM settings.
policy-cd-access	b	readwrite	Allow VM to read from CD.
policy-cd-recording	b	readwrite	Allow VM to burn CDs.

Name	Type	Access	Description
policy-audio-access	b	readwrite	Allow VM to playback audio.
policy-audio-recording	b	readwrite	Allow VM to record audio.
policy-wired-networking	b	readwrite	Allow VM to access wired network.
policy-wireless-networking	b	readwrite	Allow VM to access wireless network.
policy-print-screen	b	readwrite	Allow VM to use the PrintScreen key.
run-post-create	s	readwrite	Command to run post VM creation.
run-pre-delete	s	readwrite	Command to run pre VM deletion.
run-pre-boot	s	readwrite	Command to run synchronously pre VM boot.
run-insteadof-start	s	readwrite	Command to run instead of VM start
run-on-state-change	s	readwrite	Command to run on VM state change.
run-on-acpi-state-change	s	readwrite	Command to run on VM acpi state change.
domstore-read-access	b	readwrite	Configure domain read access to domstore.
domstore-write-access	b	readwrite	Configure domain write access to domstore.
show-switcher	b	readwrite	Hide/show switcher bar inside VM.
native-experience	b	readwrite	Toggle 'native experience' on/onff.
wireless-control	b	readwrite	Give VM control over wireless stack.
s3-mode	s	readwrite	Configure how the VM is put to sleep.
s4-mode	s	readwrite	Configure how the VM is hibernated.
vsnd	b	readwrite	Use PV audio device.
vkbd	b	readwrite	Use PV keyboard and mouse.
vfb	b	readwrite	Use PV framebuffer.
v4v	b	readwrite	Use V4V.
private-space	i	read	Private space used (in MiB).
realm	s	readwrite	Realm ID.
sync-uuid	s	readwrite	VM UUID in synchroniser space.
icbinn-path	s	readwrite	Filesystem path exported to the VM via icbinn server.

Name	Type	Access	Description
ovf-transport-iso	b	readwrite	Transport of OVF configuration via iso enabled yes/no.
download-progress	i	readwrite	VM download progress.
ready	b	readwrite	Is VM ready for use?
restrict-display-depth	b	readwrite	Restrict available display depths. Currently required by emulated VGA in Windows 8.
restrict-display-res	b	readwrite	Restrict available display resolutions.
preserve-on-reboot	b	readwrite	After reboot, keep the vm in rebooted state instead of automatically restarting it
boot-sentinel	s	readwrite	Name of xenstore node to wait on before completing vm startup from toolstack PoV. Useful if VM exports services which need to be immediately usable by other vms
hpet	b	readwrite	HPET support
timer-mode	i	readwrite	Domain timer mode
nestedhvm	b	readwrite	Enable nested virtualization
serial	s	readwrite	Serial port specification

11.47. Interface com.citrix.xenclient.xenmgr.vm.unrestricted

Allows unrestricted access to VM properties (without policy checks) so daemons in the control domain (dom0) can easily change settings.

11.47.1. Properties:

Name	Type	Access	Description
state	s	read	
acpi-state	i	read	
domid	i	read	
type	s	readwrite	
name	s	readwrite	
description	s	readwrite	
uuid	s	read	
seamless-id	s	readwrite	Handle seamless uses to reference the VMs.

Name	Type	Access	Description
slot	i	readwrite	
pv-addons	b	readwrite	
pv-addons-version	s	readwrite	
start-on-boot	b	readwrite	
start-from-suspend-image	s	readwrite	Start the VM from the suspend image file, if it exists.
time-offset	i	readwrite	
crypto-user	s	readwrite	
auto-s3-wake	b	readwrite	
os	s	readwrite	
image-path	s	readwrite	
wired-network	s	readwrite	
wireless-network	s	readwrite	
gpu	s	readwrite	
cd	s	readwrite	
mac	s	read	
amt-pt	b	readwrite	
portica-enabled	i	read	
portica-installed	b	read	
seamless-traffic	b	readwrite	
autostart-pending	b	read	
hibernated	b	read	
memory-static-max	i	readwrite	
memory-target	i	read	
memory-min	i	readwrite	
memory	i	readwrite	
hidden-in-switcher	b	readwrite	
hidden-in-ui	b	readwrite	

Name	Type	Access	Description
notify	s	readwrite	
hvm	b	readwrite	
pae	b	readwrite	
apic	b	readwrite	
viridian	b	readwrite	
cpuid	s	readwrite	Used to set/unset certain bits in response to CPUID instruction.
xci-cpuid-signature	b	readwrite	Advertise the guest as XenClient flavour of Xen.
hap	b	readwrite	
nx	b	readwrite	
sound	s	readwrite	
display	s	readwrite	
boot	s	readwrite	
cmd-line	s	readwrite	
kernel-extract	s	readwrite	[disk_number,partition_number:]/path/to/kernel Location of kernel image within the VM's disk(s). Examples: /boot/vmlinuz (/boot/vmlinuz in unpartitioned disk 0) 2:/boot/vmlinuz (/boot/vmlinuz in unpartitioned disk 2) 0,1:/boot/vmlinuz (/boot/vmlinuz in 1st partitin of disk 0)
kernel	s	readwrite	
initrd-extract	s	readwrite	[disk_number,partition_number:]/path/to/initrd Location of initrd image within the VM's disk(s). Examples: /boot/inird (/boot/inird in unpartitioned disk 0) 2:/boot/inird (/boot/inird in unpartitioned disk 2) 0,1:/boot/inird (/boot/inird in 1st partitin of disk 0)
initrd	s	readwrite	
acpi-pt	b	readwrite	Enable OEM windows installation by exposing the ACPI SLIC table.
smbios-pt	b	readwrite	
vcpus	i	readwrite	
cores-per-socket	i	readwrite	

Name	Type	Access	Description
videoram	i	readwrite	
passthrough-mmio	s	readwrite	
passthrough-io	s	readwrite	
flask-label	s	readwrite	
qemu-dm-path	s	readwrite	
qemu-dm-timeout	i	readwrite	Timeout (in seconds) to wait for qemu response during its startup.
oem-acpi-features	b	readwrite	Enables access to additional OEM acpi features.
start-on-boot-priority	i	readwrite	Control priority of start-on-boot VMs, higher priority means earlier start. For new VMs this defaults to 0.
shutdown-priority	i	readwrite	Controls order of vm shutdown, higher priority means earlier shutdown. For new VMs defaults to 0, or -10 in case of PVM. VMs with same priority are shutdown in parallel.
keep-alive	b	readwrite	Automatically restart this VM if it shuts down or crashes.
provides-network-backend	b	readwrite	Whether this domain is a networking backend and handles the physical nic devices.
provides-default-network-backend	b	readwrite	Whether this domain is the primary networking backend and handles the physical NIC devices.
provides-graphics-fallback	b	readwrite	System will keep this VM on screen when no better candidate is present.
measured	b	read	if VM is measured, the hash of the disk image is checked before it is allowed to start.
extra-xenvm	s	readwrite	Extra xenvm arguments separated by semicolon.
extra-hvm	s	readwrite	Extra ioemu arguments separated by semicolon.
crypto-key-dirs	s	readwrite	Comma-separated list of disk encryption keys directories.
dependencies	ao	read	VMs that are required to be running before starting this one.
track-dependencies	b	readwrite	If true, automatically start required VMs.
seamless-mouse-left	i	read	Where to move mouse when it goes over left edge.

Name	Type	Access	Description
seamless-mouse-right	i	read	Where to move mouse when it goes over right edge.
control-platform-power-state	b	readwrite	If set, whole platform will go into S3/S4/S5 when this VM goes into S3/S4/S5.
stubdom	b	readwrite	Use stub domain to hide the device emulator.
usb-enabled	b	readwrite	Enables PV USB support.
usb-control	b	readwrite	Enables VM to talk to USB daemon
usb-grab-devices	b	readwrite	Automatically assign all available USB devices upon this VM start.
greedy-pciback-bind	b	readwrite	Bind passthrough pci devices to pciback early, to prevent their use by other VMs.
policy-modify-vm-settings	b	readwrite	Allow modification of VM settings.
policy-cd-access	b	readwrite	Allow to read from CD.
policy-cd-recording	b	readwrite	Allow to record CD.
policy-audio-access	b	readwrite	Allow to playback audio.
policy-audio-recording	b	readwrite	Allow to record audio.
policy-wired-networking	b	readwrite	Allow to access wired network.
policy-wireless-networking	b	readwrite	Allow to access wireless network.
policy-print-screen	b	readwrite	Allow VM to use the PrintScreen key.
run-post-create	s	readwrite	Command to run post VM creation.
run-pre-delete	s	readwrite	Command to run pre VM deletion.
run-pre-boot	s	readwrite	Command to run synchronously pre VM boot.
run-insteadof-start	s	readwrite	Command to run instead of VM start
run-on-state-change	s	readwrite	Command to run on VM state change.
run-on-acpi-state-change	s	readwrite	Command to run on VM acpi state change.
domstore-read-access	b	readwrite	Configure domain read access to domstore.
domstore-write-access	b	readwrite	Configure domain write access to domstore.
show-switcher	b	readwrite	Hide/show switcher bar inside VM.
native-experience	b	readwrite	Toggle 'native experience' on/off.

Name	Type	Access	Description
wireless-control	b	readwrite	Give VM control over wireless stack.
s3-mode	s	readwrite	Configure how the VM is put to sleep.
s4-mode	s	readwrite	Configure how the VM is hibernated.
vsnd	b	readwrite	Use PV audio device.
vkbd	b	readwrite	Use PV keyboard and mouse.
vfb	b	readwrite	Use PV framebuffer.
v4v	b	readwrite	Use V4V.
private-space	i	read	Private space used (in MiB).
realm	s	readwrite	Realm ID.
sync-uuid	s	readwrite	VM UUID in Synchroniser space.
icbinn-path	s	readwrite	Filesystem path exported to the VM via icbinn server.
ovf-transport-iso	b	readwrite	Transport of OVF configuration via iso enabled yes/ no.
download-progress	i	readwrite	VM download progress.
ready	b	readwrite	Is VM ready for use?
restrict-display-depth	b	readwrite	Restrict available display depths. Currently required by emulated VGA in Windows 8.
restrict-display-res	b	readwrite	Restrict available display resolutions.
preserve-on-reboot	b	readwrite	After reboot, keep the vm in rebooted state instead of automatically restarting it
boot-sentinel	s	readwrite	Name of xenstore node to wait on before completing vm startup from toolstack PoV. Useful if VM exports services which need to be immediately usable by other vms
hpet	b	readwrite	HPET support
timer-mode	i	readwrite	Domain timer mode
nestedhvm	b	readwrite	Enable nested virtualization
serial	s	readwrite	Serial port specification

11.48. Interface com.citrix.xenclient.xenmgr.vm.product

Configuration of products (services) in the VM.

11.48.1. Methods:

11.48.1.1. Method get_ovf_env_xml

Return the OVF environment xml.

The arguments are:

Type	Name	Direction	Description
s	value	out	

11.48.1.2. Method list_product_properties

List all product properties for this VM.

The arguments are:

Type	Name	Direction	Description
aa{ss}	product_properties	out	

11.48.1.3. Method get_product_property

Query product property value.

The arguments are:

Type	Name	Direction	Description
s	property_id	in	
s	value	out	

11.48.1.4. Method set_product_property

Change product property value.

The arguments are:

Type	Name	Direction	Description
s	property_id	in	
s	value	in	

11.49. Interface com.citrix.xenclient.xenmgr.vm.auth

Authentication Interface

11.49.1. Methods:

11.49.1.1. Method auth_required

Check if authentication to VM is required.

The arguments are:

Type	Name	Direction	Description
b	required	out	

11.49.1.2. Method auth

Attempt to authenticate to VM.

This method has no arguments.

11.50. Interface com.citrix.xenclient.xenmgr.vm.pci

Manipulate passthrough lists.

11.50.1. Methods:

11.50.1.1. Method add_pt_rule

Add PCI passthrough rule. Passing empty string in one of the arguments (class,vendor,device) turns off matching on that argument.

The arguments are:

Type	Name	Direction	Description
s	pciclass	in	
s	vendor_id	in	
s	device_id	in	

11.50.1.2. Method add_pt_rule_bdf

Add PCI passthrough rule. Specify device using BDF syntax (e.g.: 0000:00:16.0).

The arguments are:

Type	Name	Direction	Description
s	bdf	in	

11.50.1.3. Method delete_pt_rule

Remove passthrough rule(s).

The arguments are:

Type	Name	Direction	Description
s	pciclass	in	
s	vendor_id	in	
s	device_id	in	

11.50.1.4. Method delete_pt_rule_bdf

Remove PCI passthrough rule using BDF syntax.

The arguments are:

Type	Name	Direction	Description
s	bdf	in	

11.50.1.5. Method list_pt_rules

List current passthrough rules in effect.

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

11.50.1.6. Method list_pt_pci_devices

List PCI devices which match current passthrough rules.

The arguments are:

Type	Name	Direction	Description
aa{ss}		out	

VM lifecycle state codes.

Managed disk type.

S3 mode.

S4 mode.

11.51. Interface com.citrix.xenclient.xenmgr

Main xenmgr interface, used for VM creation and enumeration.

11.51.1. Methods:

11.51.1.1. Method list_vms

List each VM present. Returns a list of dicts with few critical properties filled for each VM, including VM state, domain ID (if running), uuid etc.

The arguments are:

Type	Name	Direction	Description
ao	paths	out	

11.51.1.2. Method list_domids

List current domain IDs.

The arguments are:

Type	Name	Direction	Description
ai	domids	out	

11.51.1.3. Method list_child_service_vm_templates

List the templates for creating child service VMs.

The arguments are:

Type	Name	Direction	Description
as	templates	out	

11.51.1.4. Method list_templates

List the templates for creating new VMa.

The arguments are:

Type	Name	Direction	Description
as	templates	out	

11.51.1.5. Method list_ui_templates

List the UI-visible VM creation templates.

The arguments are:

Type	Name	Direction	Description
aa{ss}	templates	out	

11.51.1.6. Method list_extension_packs

List installed extension packs. Returns array of dictionaries, where each dict contains following keys: vendor, name, description, version, image.

The arguments are:

Type	Name	Direction	Description
aa{ss}	packs	out	

11.51.1.7. Method find_vm_by_uuid

Returns the object path to the VM with the given UUID, or raises an error.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	out	

11.51.1.8. Method find_vm_by_domid

Returns the object path to the VM of the given domain ID. Fails with an error if no such VM is running.

The arguments are:

Type	Name	Direction	Description
i	domid	in	
o	obj_path	out	

11.51.1.9. Method create_vm

Create a new VM.

The arguments are:

Type	Name	Direction	Description
o	path	out	

11.51.1.10. Method create_vm_with_template

Create a new VM based on the given template.

The arguments are:

Type	Name	Direction	Description
s	template	in	

Type	Name	Direction	Description
o	path	out	

11.51.1.11. Method create_vm_with_template_and_uuid

Create a new VM based on the given template and UUID.

The arguments are:

Type	Name	Direction	Description
s	template	in	
s	uuid	in	
o	path	out	

11.51.1.12. Method create_vm_with_template_and_json

Create a new VM based on given template file and JSON blob. Template or json can be left empty

The arguments are:

Type	Name	Direction	Description
s	template	in	
s	json	in	
o	path	out	

11.51.1.13. Method create_vm_with_ui

Create a new VM based on the given template, with ui properties initially set.

The arguments are:

Type	Name	Direction	Description
s	template	in	
s	name	in	
s	description	in	
s	image_path	in	
o	path	out	

11.51.1.14. Method create_vhd

Create a new VHD.

The arguments are:

Type	Name	Direction	Description
i	size_mb	in	
s	path	out	

11.51.2. Signals:

11.51.2.1. vm_config_changed

Notify that VM configuration has changed.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	in	

11.51.2.2. vm_state_changed

Notify that VM state has changed.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	in	
s	state	in	
i	acpi_state	in	

11.51.2.3. vm_name_changed

Notify that VM name has changed.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	in	

11.51.2.4. config_changed

Notify xenmgr that host level configuration has changed.

This signal has no arguments.

11.51.2.5. language_changed

Notify that the language settings have changed.

This signal has no arguments.

11.51.2.6. vm_created

Notify that a VM was created.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	in	

11.51.2.7. vm_deleted

Notify that a VM was deleted.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	in	

11.51.2.8. network_state_changed

Notify when a network becomes available/unavailable.

The arguments are:

Type	Name	Direction	Description
b	available	in	

11.51.2.9. vm_transfer_changed

VM download/upload progress has changed.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

Type	Name	Direction	Description
o	obj_path	in	

11.51.2.10. cd_assignment_changed

CD device VM assignment has changed.

The arguments are:

Type	Name	Direction	Description
s	dev	in	
s	uuid	in	
o	obj_path	in	

11.52. Interface com.citrix.xenclient.xenmgr.config

xenmgr configuration information, mostly living in the /xenmgr db tree.

11.52.1. Properties:

Name	Type	Access	Description
iso-path	s	readwrite	
autostart	b	readwrite	
pvm-autostart-delay	i	readwrite	
svm-autostart-delay	i	readwrite	
v4v-hosts-file	b	readwrite	
use-networking-domain	b	read	
bypass-sha1sum-checks	b	read	
xc-diag-timeout	i	readwrite	
platform-crypto-key-dirs	s	readwrite	Comma separated list of disk encryption keys directories.
guest-only-networking	b	read	If true, networking features in dom0/networking VM are disabled.
vm-creation-allowed	b	readwrite	Allow creation of VMs on this host.
vm-deletion-allowed	b	readwrite	Allow deletion of VM on this host.
ota-upgrades-allowed	b	readwrite	Allow over the air upgrades on this host.

Name	Type	Access	Description
connect-remote-desktop-allowed	b	readwrite	Allow using remote desktop (via icavm).
measure-fail-action	s	readwrite	Action to perform when computing service VM checksum fails: sleep,hibernate,shutdown,reboot,nothing.
v4v-firewall	b	readwrite	If true, v4v firewall is enabled, rejecting all incoming traffic by default.
secondary-gpu-pt	b	read	True if passthrough of secondary GPUs supported.
configurable-save-changes-across-reboots	b	read	True if user can opt to not save VHD changes across VM reboot.
enable-ssh	b	readwrite	Enable external SSH access to dom0.
enable-v4v-ssh	b	readwrite	Enable internal SSH access to dom0.
enable-dom0-networking	b	readwrite	Have dom0 connect to external network using DHCP.
dom0-mem-target-mib	i	readwrite	Dom0 balloon memory target, in mebibytes. If 0, don't balloon dom0.
autolock-cd-drives	b	readwrite	Automatically lock/unlock cd drives to vms on media insert/eject events

11.53. Interface com.citrix.xenclient.xenmgr.config.ui

UI configuration information.

11.53.1. Properties:

Name	Type	Access	Description
show-msg-on-vm-start	b	readwrite	
show-msg-on-vm-start-tools-warning	b	readwrite	
show-msg-on-no-disk	b	readwrite	
show-mboot-warning	b	readwrite	
show-tools-warning	b	readwrite	
wallpaper	s	readwrite	
pointer-trail-timeout	i	readwrite	
view-type	s	readwrite	

Name	Type	Access	Description
modify-settings	b	readwrite	
modify-services	b	readwrite	
modify-advanced-vm-settings	b	readwrite	
switcher-enabled	b	readwrite	True if seamless mouse switching is enabled.
switcher-self-switch-enabled	b	readwrite	If false, switching to the actual current slot will do nothing.
switcher-keyboard-follows-mouse	b	readwrite	If false, when mouse-switching, the keyboard stays in it's slot until a mouse click.
switcher-resistance	i	readwrite	Determines how hard the edges are for mouse-switching.
idle-time-threshold	i	readwrite	Determines the amount of time the host can be idle after which it goes to sleep.
language	s	readwrite	Current language.
supported-languages	as	read	List of supported languages.
drm-graphics	b	readwrite	Enable DRM graphics plugin on older hardware (pre-haswell)

11.54. Interface com.citrix.xenclient.policy

Implements the policy enforce/retrieve.

11.54.1. Methods:

11.54.1.1. Method enforce

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
s	value	in	

11.54.1.2. Method retrieve

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

Type	Name	Direction	Description
s	result	out	

11.55. Interface com.citrix.xenclient.xenmgr.diag

Helpers for gathering diagnostic info from guests, and dom0.

11.55.1. Methods:

11.55.1.1. Method save

Get diagnostic information for VMs and save to the specified directory.

The arguments are:

Type	Name	Direction	Description
s	mode	in	
s	dir	out	

11.55.1.2. Method gather

Put a file containing results of xc-diag to dom0.

The arguments are:

Type	Name	Direction	Description
s	name	in	
s	data_	in	

11.55.1.3. Method create_status_report

Create a report using status-tool.

The arguments are:

Type	Name	Direction	Description
b	screenshots	in	
b	guest_info	in	
s	summary	in	
s	description	in	
s	repro_steps	in	

Type	Name	Direction	Description
s	ticket	in	
s	file	out	

11.55.1.4. Method taas_authenticate_credentials

Validate MyCitrix credentials.

The arguments are:

Type	Name	Direction	Description
s	username	in	
s	password	in	
as	result	out	

11.55.1.5. Method taas_upload

Upload file to TaaS.

The arguments are:

Type	Name	Direction	Description
s	username	in	
s	password	in	
s	caseid	in	
s	filename	in	
b	result	out	

11.55.1.6. Method taas_agree_terms

TaaS Terms.

The arguments are:

Type	Name	Direction	Description
s	username	in	
s	password	in	
s	version	in	
b	result	out	

11.55.1.7. Method status_report_screen

Show or hide the status report screen.

The arguments are:

Type	Name	Direction	Description
b	show	in	

11.55.2. Signals:

11.55.2.1. gather_request

Sent to notify guests to start preparing diagnostics info.

The arguments are:

Type	Name	Direction	Description
s	mode	in	

11.56. Interface com.citrix.xenclient.xenmgr.testing

Helpers for automated testing.

11.56.1. Methods:

11.56.1.1. Method script_queue

Add a script to the fifo queue of test scripts to exercise in the UI.

The arguments are:

Type	Name	Direction	Description
s	script	in	

11.56.1.2. Method script_dequeue

Remove and return script.

The arguments are:

Type	Name	Direction	Description
s	script	out	

11.57. Interface com.citrix.xenclient.xenmgr.unrestricted

Allows xenmgr operation ignoring policy checks so daemons in the control domain (dom0) can easily change settings.

11.57.1. Methods:

11.57.1.1. Method unrestricted_create_vm

Create a new VM.

The arguments are:

Type	Name	Direction	Description
o	path	out	

11.57.1.2. Method unrestricted_create_vm_with_template_and_json

Create a new VM based on given template file and JSON blob. Template or json can be left empty.

The arguments are:

Type	Name	Direction	Description
s	template	in	
s	json	in	
o	path	out	

11.57.1.3. Method unrestricted_delete_vm

Delete existing VM.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	

11.58. Interface com.citrix.xenclient.xenmgr.guestreq

Some well defined requests coming from guests.

11.58.1. Methods:

11.58.1.1. Method request_attention

Guest requests user attention. Executing this method from any guest VM (User VM or Service VM) will cause a dialog box in both the UIVM and Windows guests to alert the user that the VM which sent it requires attention. This is useful for the case where a service VM needs credentials from the user or has an issue which requires user intervention to resolve. Note that the dialog box means a VM wants you to switch to the VM; but it does NOT mean that we recommend that you take any action, including but not limited to giving your attention to any VM that is requesting said attention. A malicious VM can repeatedly call this function and cause this dialog box to appear repeatedly; there is no time delay that limits how often any VM can request attention.

This method has no arguments.

11.58.2. Signals:

11.58.2.1. requested_attention

Notify that VM has requested attention.

The arguments are:

Type	Name	Direction	Description
s	uuid	in	
o	obj_path	in	

11.59. Interface xenvm.signal.notify

11.59.1. Signals:

11.59.1.1. notify

Signals a VM state change.

The arguments are:

Type	Name	Direction	Description
as	vmstate	in	

Chapter 12. Example OpenXT D-Bus Application

This example application is a simple script to query the OpenXT toolstack for two simple VM properties. It begins by enumerating the object paths of all VMs installed on the platform. It then queries two properties from the `com.citrix.xenclient.xenmgr.vm` interface for each of these VM objects. First the VM `name` property is queried, then the VM `state` property. The output is formatted to display the VM object path, the VM `name` and the current state of the VM.

```
#!/bin/sh

REGX_OBJ='s&^[[:space:]]\+object[[:space:]]\+path[[:space:]]\+"\(.*\)"[[:space:]]*$&\1&p'
REGX_STR='s&^[[:space:]]\+variant[[:space:]]\+string[[:space:]]\+\"\(.*\)\"[[:space:]]*$&\1&p'

dbus-send --system --print-reply \ ❶
    --dest=com.citrix.xenclient.xenmgr \ ❷
    / \ ❸
    com.citrix.xenclient.xenmgr.list_vms | \
    sed -n "${REGX_OBJ}" | \
    while read VM; do
        echo -n "The VM object ${VM} named \""
        dbus-send --system --print-reply \
            --dest=com.citrix.xenclient.xenmgr \
            ${VM} \ ❹
            org.freedesktop.DBus.Properties.Get \
            string:'com.citrix.xenclient.xenmgr.vm' \
            string:'name' | \
            sed -n "${REGX_STR}" | \
            tr -d '\n'
        echo -n "\" is in the \""
        dbus-send --system --print-reply \
            --dest=com.citrix.xenclient.xenmgr \
            ${VM} \
            org.freedesktop.DBus.Properties.Get \
            string:'com.citrix.xenclient.xenmgr.vm' \
            string:'state' | \
            sed -n "${REGX_STR}" | \
            tr -d '\n'
      echo "\" state" ❺
done
```

❶ Connect to the system bus.
❷ Obtain a reference to the xenmgr service.
❸ Access and introspect the root object.
❹ Access and introspect VM objects.
❺ Print the name and current state of all VMs.

The script above assumes that it is being run from within the control domain. It is possible for the script to be run from a service VM but its environment must be set up to cause the DBus client to connect to the bus over V4V. Assuming the script above is named `list-vms.sh` we can simply wrap it in another script which sets the necessary environment variables.

```
#!/bin/sh

DBUS_SYSTEM_BUS_ADDRESS='tcp:host=1.0.0.0,port=5556' \
LD_PRELOAD='/usr/lib/libv4v-1.0.so.0' \
INET_IS_V4V='1' \
list-vms.sh
```

For this script to function properly your service VM must be configured according to the instructions in Section 3.1: “Creating a service VM”. Further, your must have the `dbus-bouncer` configured and running. By default, OpenXT allows guests to query for basic VM properties over DBUS to support the in-guest switcher. Additional DBUS functionality requires the administrator to explicitly allow the service VM access to the appropriate objects and methods.

Chapter 13. VM Database Configuration File Syntax

The properties associated with each virtual machine are defined in a human-readable text file. This file is managed by the OpenXT tool stack and is not typically edited by hand. If you edit this file manually, run the command `killall HUP dbd` in the control domain console window to signal the `dbd` process to reload the database. The following sections explain the syntax of an example VM database file.

13.1. VM Identity and General Configuration

```
{
  "uuid": "3a9ecb37-a563-45ca-ad06-f4f313e202f4", ❶
  "type": "svm", ❷
  "image_path": "images\/vms\/001_ComputerWin7_h32bit_120.png", ❸
  "config": { ❹
    "notify": "dbus", ❺
    "hvm": "true", ❻
    "pae": "true", ❼
    "acpi": "true", ❽
    "apic": "true", ❾
    "viridian": "true",
    "hap": "true",
    "nx": "true",
    "sound": "ac97", ❿
    "memory": "1024", ⓫
    "display": "none", ⓬
    "boot": "cd", ⓭
    "extra-xenvm": { ⓮
      "0": "flask-label=system_u:system_r:domU_t" ⓯
    },
```

❶ OpenXT ID of the VM.
❷ The VM type (svm|pvm). pvm indicates that the VM has 3D Graphics Support enabled
❸ The path to the VM icon file.
❹ VM configuration section.
❺ VM configuration section.
❻ Is the VM an HVM? (true/false)
❼ Use PAE? (true/false)
❽ Use ACPI? (true/false)
❾ Use APIC? (true/false)
❿ Sound card to emulate in the VM
⓫ Quantity of RAM allocated to the VM
⓬ Display type to use
⓭ Boot order (c for hard drive, d for cd and n for network)
⓮ Extra options
⓯ VM label

13.2. VM Networking

```
"nic": { ❶
  "0": { ❷
    "id": "0", ❸
    "bridge": "brbridged", ❹
    "firewall-rules": { ❺
      "0": { ❻
        "type": "output", ❼
        "protocol": "tcp", ❽
        "ip": "10.80.248.206", ❾
        "port": "80", ❿
        "cmd": "drop" ⓫
      },
      "1": { ⓬
        "type": "output",
        "cmd": "reject"
      }
    }
  },
  "1": { ⓭
    "id": "1",
    "bridge": "brwireless"
  }
},
```

❶ Emulated network cards section begins
❷ First NIC details begin
❸ Card ID
❹ Card bridge
❺ VM-specific firewall rules
❻ Rule ID
❼ Rule type (input/output)
❽ Rule protocol (tcp/udp etc.)
❾ IP address
❿ Port
⓫ Command (accept|drop|reject)
⓬ Another VM-specific firewall rule
⓭ Another emulated network card

13.3. VM Disks and VCPUs

```
   "disk": { ❶
     "0": { ❷
       "path": "\/storage\/isos\/xc-tools.iso", ❸
       "type": "file", ❹
       "mode": "r", ❺
       "device": "hdc", ❻
       "devtype": "cdrom", ❼
       "snapshot": "" ❽
     },
     "1": { ❾
       "path": "\/storage\/disks\/d1cb870c-7cf5-46c9-a002-c00542b11b64.vhd",
       "type": "vhd",
       "mode": "w",
       "device": "hda",
       "devtype": "disk",
       "snapshot": ""
     }
   },
   "vcpus": "1" ❿
 },
```

❶ VM disks
❷ First disk
❸ Path to the virtual disk ISO, VHD file, or physical partition
❹ Virtual disk type (file|phys)
❺ Access rights (r|w)
❻ Device identifier
❼ Disk type (disk|cdrom)
❽ Path to disk snapshot
❾ Second virtual disk
❿ The number of VCPUs assigned to the VM

13.4. VM Identity and Further Settings

```
  "name": "Win7", ❶
  "slot": "1", ❷
  "description": "", ❸
  "start_on_boot": "false", ❹
  "hibernated": "false", ❺
  "time-offset": "0", ❻
  "pv-addons-installed": "true" ❼
}
```

❶ The name of the VM as displayed in UIVM for OpenXT The slot
❷ number of the VM
❸ VM description
❹ Boot the VM when booting the platform? (true|false)
❺ Is the VM hibernating? (true|false)
❻ Number of seconds to offset the VM clock
❼ Are the OpenXT Tools installed? (true|false)

Where Can I Find ...

Item	Location	Description
API and Developer Documentation	`XTEngineDeveloper_Guide.pdf`	How to extend OpenXT, create a Service VM, alter the OpenXT UI, communicate between domains, details of OVF support and full API documentation.
Sample OVF files	In `openxt-sdk.tar.gz` at `openxt-sdk/Service_VM/Packaging/sample.ovf` and `squeeze-hvm.ovf`	Sample VM appliance definition
V4V for Windows Guests	In `openxt-sdk.tar.gz` at `openxt-sdk/Service_VM/MS_Windows/Samples`	Sample code to use V4V from Windows
V4V for Linux Guests	In `openxt-sdk.tar.gz` at `openxt-sdk/Service_VM/Linux/Samples`	Sample code to use V4V from Linux
Debian OpenXT Tools	`xctools-debian-repo-ia32.tar.gz`	Debian 32-bit binary packages for XC Tools
Code to access Xen platform functionality from kernel mode in Windows	In `openxt-sdk.tar.gz` at `openxt-sdk/Service_VM/Linux/Samples`	See `openxt-sdk/Service_VM/Linux/Samples/README.txt`
Debian OpenXT Tools Source	`xctools-debian-repo-noarch.tar.gz`	Debian source packages for XC Tools
`gen-packages.sh`	In `openxt-sdk.tar.gz` at `openxt-sdk/Custom_OTA_Update`	See Chapter 2: "*Manually Adding Extension Packs to a OpenXT Installation*"
`gen-repository.sh`	In `openxt-sdk.tar.gz` at `openxt-sdk/Custom_OTA_Update`	See Chapter 2: "*Manually Adding Extension Packs to a OpenXT Installation*"
`gen-signature.sh`	In `openxt-sdk.tar.gz` at `openxt-sdk/Custom_OTA_Update`	See Chapter 2: "*Manually Adding Extension Packs to a OpenXT Installation*"
User Interface Extension Files	In `openxt-sdk.tar.gz` at `openxt-sdk/User_Interface/assets.zip` and `plugin_example.zip`	Sample logo, wallpaper, VM icon