NAME

virt−p2v − Convert a physical machine to use KVM

SYNOPSIS

```
virt-p2v

virt-p2v.iso
```

DESCRIPTION

Virt−p2v converts a physical machine to run virtualized on KVM, managed by libvirt, OpenStack, oVirt, Red Hat Virtualisation (RHV), or one of the other targets supported by **virt−v2v** (1).

Normally you don't run the virt−p2v program directly. Instead you have to boot the physical machine using the bootable CD-ROM, ISO or PXE image. This bootable image contains the virt−p2v binary and runs it automatically. Booting from a CD−ROM/etc is required because the disks which are being converted must be quiescent. It is not safe to try to convert a running physical machine where other programs may be modifying the disk content at the same time.

This manual page documents running the virt−p2v program. To create the bootable image you should look at **virt−p2v−make−disk** (1) or **virt−p2v−make−kickstart** (1).

NETWORK SETUP

Virt−p2v runs on the physical machine which you want to convert. It has to talk to another server called the ‘‘conversion server’’ which must have **virt−v2v** (1) installed on it. It always talks to the conversion server over SSH:

```
| virt-p2v     |                  | virt-v2v       |
| (physical    |  ssh connection  | (conversion    |
|  server)    server)       |
```

The virt−v2v program on the conversion server does the actual conversion (physical to virtual, and virtual to virtual conversions are sufficiently similar that we use the same program to do both).

The SSH connection is always initiated from the physical server. All data is transferred over the SSH connection. In terms of firewall and network configuration, you only need to ensure that the physical server has access to a port (usually TCP port 22) on the conversion server. Note that the physical machine may reconnect several times during the conversion process.

The reverse port forwarding feature of ssh (ie. `ssh -R`) is required by virt−p2v, and it will not work if this is disabled on the conversion server. (`AllowTcpForwarding` must be `yes` in the **sshd_config** (5) file on the conversion server).

The scp (secure copy) feature of ssh is required by virt−p2v so it can send over small files (this is *not* the method by which disks are copied).

The conversion server does not need to be a physical machine. It could be a virtual machine, as long as it has sufficient memory and disk space to do the conversion, and as long as the physical machine can connect directly to its SSH port. (See also ‘‘Resource requirements’’ in **virt−v2v** (1)).

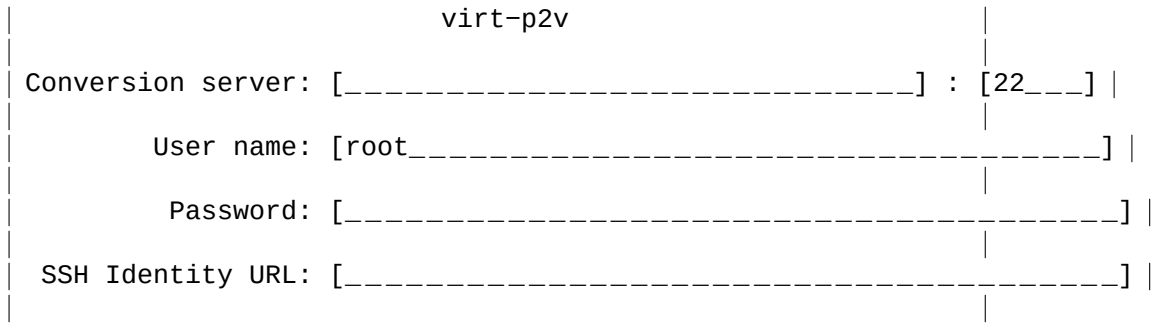
Because all of the data on the physical server's hard drive(s) has to be copied over the network, the speed of conversion is largely determined by the speed of the network between the two machines.

GUI INTERACTIVE CONFIGURATION

When you start virt−p2v, you'll see a graphical configuration dialog that walks you through connection to the conversion server, asks for the password, which local hard disks you want to convert, and other things like the name of the guest to create and the number of virtual CPUs to give it.

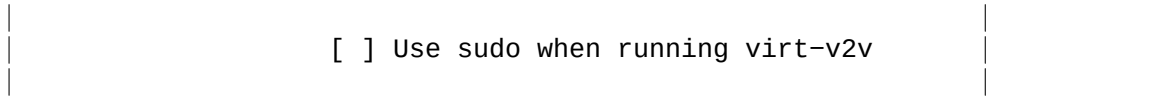
SSH CONFIGURATION DIALOG

When virt−p2v starts up in GUI mode, the first dialog looks like this:



In the fields above, you must enter the details of the conversion server: the hostname, SSH port number, remote user name, and either the password or SSH identity (private key) URL. The conversion server must have an up to date version of virt−v2v.

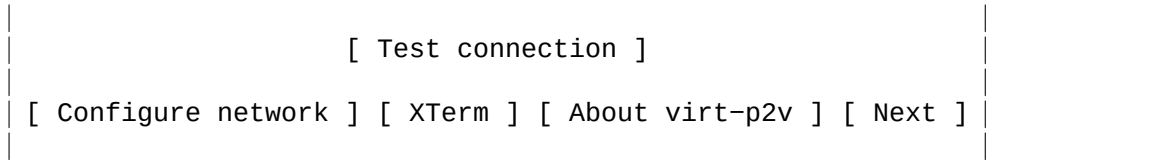
Normally you must log in to the conversion server as root, but if you check the following box:



then you can log in as another user, and virt−p2v will use the **sudo** (8) command to elevate privileges to root. Note that sudo must not require a password.

It is also possible to run virt−v2v on the conversion server entirely as non-root, but output modes may be limited. Consult the**virt−v2v** (1) manual page for details.

At the bottom of the dialog are these buttons:



You must press the `Test connection` button first to test the SSH connection to the conversion server. If that is successful (ie. you have supplied the correct server name, user name, password, etc., and a suitable version of virt−v2v is available remotely) then press the `Next` button to move to the next dialog.

You can use the `Configure network` button if you need to assign a static IP address to the physical machine, or use Wifi, bonding or other network features.

The `XTerm` button opens a shell which can be used for diagnostics, manual network configuration, and so on.

DISK AND NETWORK CONFIGURATION DIALOG

The second configuration dialog lets you configure the details of conversion, including what to convert and where to send the guest.

In the left hand column, starting at the top, the target properties let you select the name of the guest (ie. after conversion) and how many virtual CPUs and how much RAM to give it. The defaults come from the physical machine, and you can usually leave them unchanged:

```
Target properties:

       Name: [hostname_______________]

    # vCPUs: [4_____________________]

Memory (MB): [16384_________________]
```

The second panel on the left controls the virt−v2v output options. To understand these options it is a really good idea to read the **virt−v2v** (1) manual page. You can leave the options at the default to create a guest as a disk image plus libvirt XML file located in */var/tmp* on the conversion host. This is a good idea if you are a first-time virt−p2v user.

```

Virt−v2v output options:

         Output to (−o): [local            ]

     Output conn. (−oc): [___________________]

   Output storage (−os): [/var/tmp___________]

    Output format (−of): [___________________]

Output allocation (−oa): [sparse           ]
```

All output options and paths are relative to the conversion server (*not* to the physical server).

Finally in the left hand column is an information box giving the version of virt−p2v (on the physical server) and virt−v2v (on the conversion server). You should supply this information when reporting bugs.

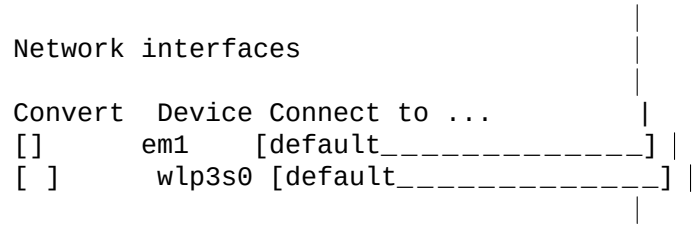
In the right hand column are three panels which control what hard disks, removable media devices, and network interfaces, will be created in the output guest. Normally leaving these at the default settings is fine.

```
Fixed hard disks

Convert  Device
[]      sda
         1024G HITACHI
         s/n 12345
[]      sdb
         119G HITACHI
         s/n 12346
```

Normally you would want to convert all hard disks. If you want virt−p2v to completely ignore a local hard disk, uncheck it. The hard disk that contains the operating system must be selected. If a hard disk is part of a RAID array or LVM volume group (VG), then either all hard disks in that array/VG must be selected, or none of them.

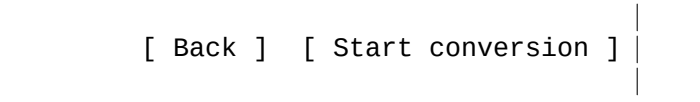
If the physical machine has CD or DVD drives, then you can use the Removable media panel to create corresponding drives on the guest after conversion. Note that any data CDs/DVDs which are mounted in the drives are *not* copied over.



In the Network interfaces panel, select the network interfaces that should be created in the guest after conversion. You can also connect these to target hypervisor networks (for further information about this feature, see ‘‘Networks and bridges’’ in **virt−v2v** (1)).

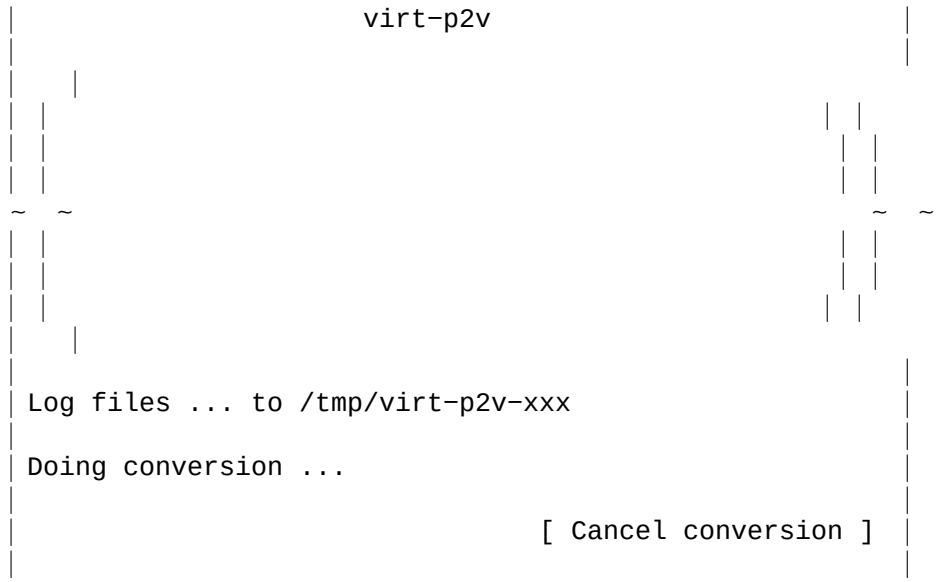
On supported hardware, left-clicking on the device name (eg. `em1`) causes a light to start flashing on the physical interface, allowing the interface to be identified by the operator.

When you are ready to begin the conversion, press the `Start conversion` button:



CONVERSION RUNNING DIALOG

When conversion is running you will see this dialog:



In the main scrolling area you will see messages from the virt−v2v process.

Below the main area, virt−p2v shows you the location of the directory on the conversion server that

contains log files and other debugging information. Below that is the current status and a button for cancelling conversion.

Once conversion has finished, you should shut down the physical machine. If conversion is successful, you should never reboot it.

KERNEL COMMAND LINE CONFIGURATION

If you don't want to configure things using the graphical UI, an alternative is to configure through the kernel command line. This is especially convenient if you are converting a lot of physical machines which are booted using PXE.

Where exactly you set command line arguments depends on your PXE implementation, but for pxelinux you put them in the `APPEND` field in the *pxelinux.cfg* file. For example:

```
DEFAULT p2v
TIMEOUT 20
PROMPT 0
LABEL p2v
  KERNEL vmlinuz0
  APPEND initrd=initrd0.img [....] p2v.server=conv.example.com p2v.password=secr
```

You have to set some or all of the following command line arguments:

p2v.remote.server=SERVER
p2v.server=SERVER

The name or IP address of the conversion server.

This is always required if you are using the kernel configuration method. If virt−p2v does not find this on the kernel command line then it switches to the GUI (interactive) configuration method.

p2v.remote.port=PORT
p2v.port=PORT

The SSH port number on the conversion server (default: `22`).

p2v.auth.username=USERNAME
p2v.username=USERNAME

The SSH username that we log in as on the conversion server (default: `root`).

p2v.auth.password=PASSWORD
p2v.password=PASSWORD

The SSH password that we use to log in to the conversion server.

The default is to try with no password. If this fails then virt−p2v will ask the user to type the password (probably several times during conversion).

This setting is ignored if `p2v.auth.identity.url` is present.

p2v.auth.identity.url=URL
p2v.identity=URL

Provide a URL pointing to an SSH identity (private key) file. The URL is interpreted by **curl** (1) so any URL that curl supports can be used here, including `https://` and `file://`. For more information on using SSH identities, see ''SSH IDENTITIES'' below.

If `p2v.auth.identity.url` is present, it overrides `p2v.auth.password`. There is no fallback.

p2v.auth.sudo
p2v.sudo

Use `p2v.sudo` to tell virt−p2v to use **sudo** (8) to gain root privileges on the conversion server after logging in as a non-root user (default: do not use sudo).

p2v.guestname=GUESTNAME

p2v.name=GUESTNAME
The name of the guest that is created. The default is to try to derive a name from the physical machine's hostname (if possible) else use a randomly generated name.

p2v.vcpus=N
The number of virtual CPUs to give to the guest. The default is to use the same as the number of physical CPUs.

p2v.memory=n(M|G)
The size of the guest memory. You must specify the unit such as megabytes or gigabytes by using for example `p2v.memory=1024M` or `p2v.memory=1G`.

The default is to use the same amount of RAM as on the physical machine.

p2v.cpu.vendor=VENDOR
The vCPU vendor, eg. ''Intel'' or ''AMD''. The default is to use the same CPU vendor as the physical machine.

p2v.cpu.model=MODEL
The vCPU model, eg. ''IvyBridge''. The default is to use the same CPU model as the physical machine.

p2v.cpu.sockets=N
Number of vCPU sockets to use. The default is to use the same as the physical machine.

p2v.cpu.cores=N
Number of vCPU cores to use. The default is to use the same as the physical machine.

p2v.cpu.threads=N
Number of vCPU hyperthreads to use. The default is to use the same as the physical machine.

p2v.cpu.acpi
Whether to enable ACPI in the remote virtual machine. The default is to use the same as the physical machine.

p2v.cpu.apic
Whether to enable APIC in the remote virtual machine. The default is to use the same as the physical machine.

p2v.cpu.pae
Whether to enable PAE in the remote virtual machine. The default is to use the same as the physical machine.

p2v.rtc.basis=(unknown|utc|localtime)
Set the basis of the Real Time Clock in the virtual machine. The default is to try to detect this setting from the physical machine.

p2v.rtc.offset=[+|−]HOURS
The offset of the Real Time Clock from UTC. The default is to try to detect this setting from the physical machine.

p2v.disks=sda,sdb,...
A list of physical hard disks to convert, for example:

```
p2v.disks=sda,sdc
```

The default is to convert all local hard disks that are found.

p2v.removable=sra,srb,...
A list of removable media to convert. The default is to create virtual removable devices for every physical removable device found. Note that the content of removable media is never copied over.

p2v.interfaces=em1,...
A list of network interfaces to convert. The default is to create virtual network interfaces for every physical network interface found.

p2v.network_map=interface:target,...
p2v.network=interface:target,...
Controls how network interfaces are connected to virtual networks on the target hypervisor. The default is to connect all network interfaces to the target `default` network.

You give a comma-separated list of `interface:target` pairs, plus optionally a default target. For example:

```
p2v.network=em1:ovirtmgmt
```

maps interface `em1` to target network `ovirtmgmt`.

```
p2v.network=em1:ovirtmgmt,em2:management,other
```

maps interface `em1` to `ovirtmgmt`, and `em2` to `management`, and any other interface that is found to `other`.

p2v.output.type=(libvirt|local|...)
p2v.o=(libvirt|local|...)
Set the output mode. This is the same as the virt−v2v *−o* option. See ‘‘OPTIONS’’ in **virt−v2v**(1).

If not specified, the default is `local`, and the converted guest is written to */var/tmp*.

p2v.output.allocation=(none|sparse|preallocated)
p2v.oa=(none|sparse|preallocated)
Set the output allocation mode. This is the same as the virt−v2v *−oa* option. See ‘‘OPTIONS’’ in **virt−v2v**(1).

p2v.output.connection=URI
p2v.oc=URI
Set the output connection libvirt URI. This is the same as the virt−v2v−*oc* option. See ‘ ‘OPTIONS’’ in **virt−v2v**(1) and http://libvirt.org/uri.html

p2v.output.format=(raw|qcow2|...)
p2v.of=(raw|qcow2|...)
Set the output format. This is the same as the virt−v2v *−of* option. See ‘‘OPTIONS’’ in **virt−v2v**(1).

p2v.output.storage=STORAGE
p2v.os=STORAGE
Set the output storage. This is the same as the virt−v2v *−os* option. See ‘‘OPTIONS’’ in **virt−v2v**(1).

If not specified, the default is */var/tmp* (on the conversion server).

p2v.pre=COMMAND
p2v.pre=‘‘COMMAND ARG ...’’
Select a pre-conversion command to run. Any command or script can be specified here. If the command contains spaces, you must quote the whole command with double quotes. The default is not to run any command.

p2v.post=poweroff
p2v.post=reboot
p2v.post=COMMAND
p2v.post=‘‘COMMAND ARG ...’’
Select a post-conversion command to run if conversion is successful. This can be any command or script. If the command contains spaces, you must quote the whole command with double quotes.

If virt−p2v is running as root, *and* the command line was set from */proc/cmdline* (not *−−cmdline*), then the default is to run the **poweroff**(8) command. Otherwise the default is not to run any command.

p2v.fail=COMMAND

p2v.fail="COMMAND ARG ..."
Select a post-conversion command to run if conversion fails. Any command or script can be specified here. If the command contains spaces, you must quote the whole command with double quotes. The default is not to run any command.

ip=dhcp
Use DHCP for configuring the network interface (this is the default).

SSH IDENTITIES

As a somewhat more secure alternative to password authentication, you can use an SSH identity (private key) for authentication.

First create a key pair. It must have an empty passphrase:

```
ssh-keygen -t rsa -N '' -f id_rsa
```

This creates a private key (`id_rsa`) and a public key (`id_rsa.pub`) pair.

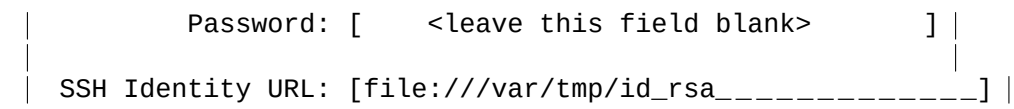
The public key should be appended to the `authorized_keys` file on the virt-v2v conversion server (usually to `/root/.ssh/authorized_keys`).

For distributing the private key, there are four scenarios from least secure to most secure:

1. Not using SSH identities at all, ie. password authentication.

   Anyone who can sniff the PXE boot parameters from the network or observe the password some other way can log in to the virt-v2v conversion server.

2. SSH identity embedded in the virt-p2v ISO or disk image. In the GUI, use:

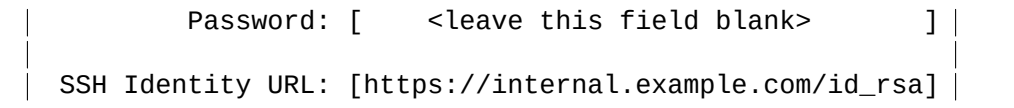
   

   or on the kernel command line:

   ```
   p2v.identity=file:///var/tmp/id_rsa
   ```

   The SSH private key can still be sniffed from the network if using standard PXE.

3. SSH identity downloaded from a website. In the GUI, use:

   or on the kernel command line:

   ```
   p2v.identity=https://internal.example.com/id_rsa
   ```

   Anyone could still download the private key and use it to log in to the virt-v2v conversion server, but you could provide some extra security by configuring the web server to only allow connections from P2V machines.

   Note that **ssh-keygen** (1) creates the `id_rsa` (private key) file with mode 0600. If you simply copy the file to a webserver, the webserver will not serve it. It will reply with "403 Forbidden" errors. You will need to change the mode of the file to make it publicly readable, for example by using:

   ```
   chmod 0644 id_rsa
   ```

4. SSH identity embedded in the virt-p2v ISO or disk image (like 2.), *and* use of secure PXE, PXE over separate physical network, or sneakernet to distribute virt-p2v to the physical machine.

Both **virt-p2v-make-disk** (1) and **virt-p2v-make-kickstart** (1) have the same option *--inject-ssh-identity* for injecting the private key into the virt-p2v disk image / ISO. See also the following manual sections:

"ADDING AN SSH IDENTITY" in **virt-p2v-make-disk** (1)

‘‘ADDING AN SSH IDENTITY’’ in **virt−p2v−make−kickstart** (1)

COMMON PROBLEMS

Timeouts

As described below (see ‘‘HOW VIRT−P2V WORKS’’) virt−p2v makes several long-lived ssh connections to the conversion server. If these connections time out then virt−p2v will fail.

To test if a timeout might be causing problems, open an XTerm on the virt−p2v machine, `ssh root@`*`conversion−server`*, and leave it for at least an hour. If the session disconnects without you doing anything, then there is a timeout which you should turn off.

Timeouts happen because:

`TIMEOUT` or `TMOUT` environment variable
: Check if one of these environment variables is set in the root shell on the conversion server.

sshd `ClientAlive*` setting
: Check for `ClientAlive*` settings in `/etc/ssh/sshd_config` on the conversion server.

Firewall or NAT settings
: Check if there is a firewall or NAT box between virt−p2v and the conversion server, and if this firewall drops idle connections after a too-short time.

  virt−p2v ≥ 1.36 attempts to work around firewall timeouts by sending ssh keepalive messages every 5 minutes.

OPTIONS

−−help
: Display help.

−−cmdline=CMDLINE
: This is used for debugging. Instead of parsing the kernel command line from */proc/cmdline*, parse the string parameter `CMDLINE`.

−−colors
−−colours
: Use ANSI colour sequences to colourize messages. This is the default when the output is a tty. If the output of the program is redirected to a file, ANSI colour sequences are disabled unless you use this option.

−−iso
: This flag is passed to virt−p2v when it is launched inside the virt−p2v ISO environment, ie. when it is running on a real physical machine (and thus not when testing). It enables various dangerous features such as the Shutdown popup button.

−−nbd=server[,server...]
: Select which NBD server is used. By default the following servers are checked and the first one found is used: *−−nbd=qemu−nbd,qemu−nbd−no−sa,nbdkit,nbdkit−no−sa*

  qemu-nbd
  : Use qemu-nbd.

  qemu-nbd-no-sa
  : Use qemu-nbd, but disable socket activation.

  nbdkit
  : Use nbdkit with the file plugin (see: **nbdkit−file−plugin** (1)).

  nbdkit-no-sa
  : Use nbdkit, but disable socket activation

  The `*−no−sa` variants allow virt−p2v to fall back to older versions of qemu-nbd and nbdkit which did not support socket activation.

−−test−disk=/PATH/TO/DISK.IMG
For testing or debugging purposes, replace */dev/sda* with a local file. You must use an absolute path.

−v
−−verbose
In libguestfs ≥ 1.33.41, debugging is always enabled on the conversion server, and this option does nothing.

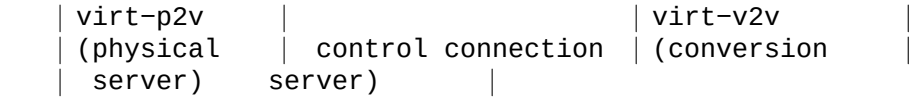
−V
−−version
Display version number and exit.

HOW VIRT−P2V WORKS

Note this section is not normative. We may change how virt−p2v works at any time in the future.

As described above, virt−p2v runs on a physical machine, interrogates the user or the kernel command line for configuration, and then establishes one or more ssh connections to the virt−v2v conversion server. The ssh connections are interactive shell sessions to the remote host, but the commands sent are generated entirely by virt−p2v itself, not by the user. For data transfer, virt−p2v will use the reverse port forward feature of ssh (ie. `ssh -R`).

It will first make one or more test connections, which are used to query the remote version of virt−v2v and its features. The test connections are closed before conversion begins.



Once virt−p2v is ready to start conversion, it will open a single ssh control connection. It first sends a mkdir command to create a temporary directory on the conversion server. The directory name is randomly chosen and is displayed in the GUI. It has the form:

```
/tmp/virt−p2v−YYYYMMDD−XXXXXXXX
```

where `YYYYMMDD` is the current date, and the 'X's are random characters.

Into this directory are written various files which include:

dmesg
lscpu
lspci
lsscsi
lsusb
(before conversion)

The output of the corresponding commands (ie **dmesg** (1), **lscpu** (1) etc) on the physical machine.

The dmesg output is useful for detecting problems such as missing device drivers or firmware on the virt−p2v ISO. The others are useful for debugging novel hardware configurations.

environment
(before conversion)

The content of the environment where **virt−v2v** (1) will run.

name
(before conversion)

The name (usually the hostname) of the physical machine.

physical.xml
(before conversion)

Libvirt XML describing the physical machine. It is used to pass data about the physical source host to **virt-v2v**(1) via the *-i libvirtxml* option.

Note this is not "real" libvirt XML (and must **never** be loaded into libvirt, which would reject it anyhow). Also it is not the same as the libvirt XML which virt-v2v generates in certain output modes.

p2v-version
v2v-version
(before conversion)

The versions of virt-p2v and virt-v2v respectively.

status
(after conversion)

The final status of the conversion. `0` if the conversion was successful. Non-zero if the conversion failed.

time
(before conversion)

The start date/time of conversion.

virt-v2v-conversion-log.txt
(during/after conversion)

The conversion log. This is just the output of the virt-v2v command on the conversion server. If conversion fails, you should examine this log file, and you may be asked to supply the **complete**, **unedited** log file in any bug reports or support tickets.

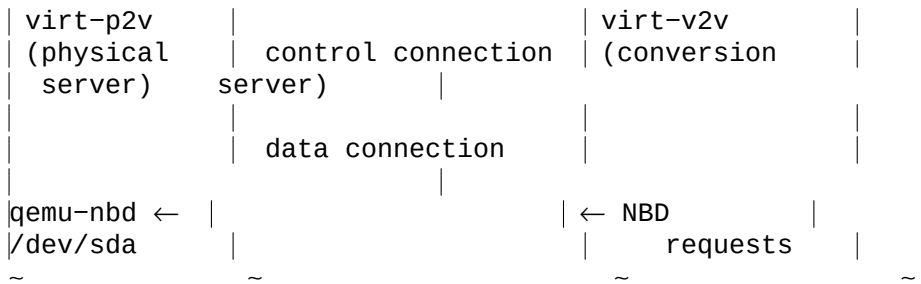
virt-v2v-wrapper.sh
(before conversion)

This is the wrapper script which is used when running virt-v2v. For interest only, do not attempt to run this script yourself.

Before conversion actually begins, virt-p2v then makes one or more further ssh connections to the server for data transfer.

The transfer protocol used currently is NBD (Network Block Device), which is proxied over ssh. The NBD server is **qemu-nbd**(1) by default but others can be selected using the *--nbd* command line option.

There is one ssh connection per physical hard disk on the source machine (the common case — a single hard disk — is shown below):



Although the ssh data connection is originated from the physical server and terminates on the conversion server, in fact NBD requests flow in the opposite direction. This is because the reverse port forward feature of ssh (`ssh -R`) is used to open a port on the loopback interface of the conversion server which is proxied back by ssh to the NBD server running on the physical machine. The effect is that virt-v2v via libguestfs can open nbd connections which directly read the hard disk(s) of the physical server.

Two layers of protection are used to ensure that there are no writes to the hard disks: Firstly, the qemu-nbd

-r (readonly) option is used. Secondly libguestfs creates an overlay on top of the NBD connection which stores writes in a temporary file on the conversion file.

The long `virt-v2v-ilibvirtxmlphysical.xml...` command is wrapped inside a wrapper script and uploaded to the conversion server. The final step is to run this wrapper script, in turn running the virt-v2v command. The virt-v2v command references the *physical.xml* file (see above), which in turn references the NBD listening port(s) of the data connection(s).

Output from the virt-v2v command (messages, debugging etc) is saved both in the log file on the conversion server. Only informational messages are sent back over the control connection to be displayed in the graphical UI.

SEE ALSO

virt-p2v-make-disk (1), **virt-p2v-make-kickstart** (1), **virt-p2v-make-kiwi** (1), **virt-v2v** (1), **qemu-nbd** (1), **nbdkit** (1), **nbdkit-file-plugin** (1), **ssh** (1), **sshd** (8), **sshd_config** (5), http://libguestfs.org/.

AUTHORS

Matthew Booth

John Eckersberg

Richard W.M. Jones http://people.redhat.com/~rjones/

Mike Latimer

Pino Toscano

Tingting Zheng

COPYRIGHT

LICENSE

BUGS

To get a list of bugs against libguestfs (which include virt-p2v), use this link: https://bugzilla.redhat.com/buglist.cgi?component=libguestfs&product=Virtualization+Tools

To report a new bug against libguestfs, use this link: https://bugzilla.redhat.com/enter_bug.cgi?component=libguestfs&product=Virtualization+Tools

When reporting a bug, please supply:

- The version of virt-p2v.
- Where you got virt-p2v (eg. which Linux distro, compiled from source, etc)
- Describe the bug accurately and give a way to reproduce it.