Converting, Inspecting, & Modifying Virtual Machines with

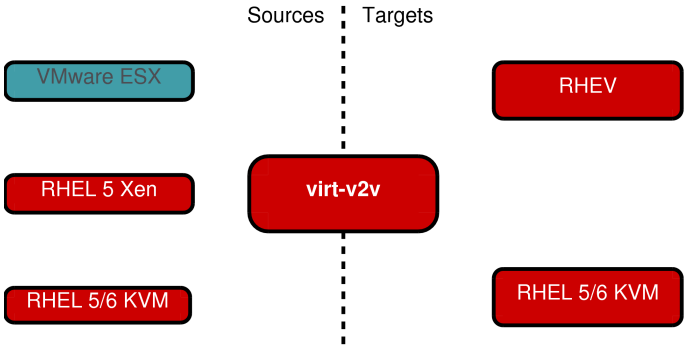
Red Hat Enterprise Linux 6.1

Matthew Booth <mbooth@redhat.com>

Red Hat Summit 2011

Converting workloads from virtual environments

Since its release, Red Hat Enterprise Linux 6 has included virt-v2v, a command line tool to manage the conversion of virtual machine images between hypervisors. The tool supports the conversion of guests running Red Hat Enterprise Linux, Fedora or Microsoft Windows between the VMware ESX, Red Hat Enterprise Linux and Red Hat Enterprise Virtualization.



It is primarily documented in its man pages:

- virt-v2v(1)
- virt-v2v.conf(5)

and the V2V Guide, which can be found linked from the URL below:

- http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/index.html.

Conversion Process

The conversion process is driven by virt-v2v, meaning there is no need to do anything to the source guest other than shut it down during conversion. The process is:

- Copy storage from the source hypervisor.
- Perform any necessary conversion on the guest operating system for its target environment.
- Create a new guest on the target hypervisor with the copied and converted storage.

The guest conversion process depends on the guest operating system. In general, the aim is to make the minimum number of modifications to ensure correct and reliable operation in the new environment. Virt-v2v will also automatically enable VirtIO, KVM's paravirtualized block and network drivers, if the guest operating system supports it.

The changes made to a guest operating system are summarised below:

- *Red Hat Enterprise Linux/Fedora*
  - Remove foreign hypervisor-specific software, e.g. Xen kmods and VMware Tools.
  - Install a new kernel if the current isn't
- *Microsoft Windows*
  - Install VirtIO drivers in the guest.
  - Add the VirtIO block drivers to the Critical Device Database.
  - Install the Red Hat Enterprise

bootable or doesn't support VirtIO.
- Configure X and text consoles.
- Update references to block devices whose names have changed.
- Update block and network drivers.
- Configure the system to boot using the relevant block drivers.

Virtualization software update agent.

Note that as we copy storage first, we never modify the original guest.

Running virt-v2v

Virt-v2v is a command-line program. Below is an example command line and its output, showing conversion of an ESX guest to run on Red Hat Enterprise Virtualization:

```
# virt-v2v -o rhev \
     -os qnap.rhev.marston:/rhev-export1 \
     -of raw -oa preallocated \
     --network rhevm \
     -ic esx://yellow.rhev.marston/?no_verify=1 RHEL4
RHEL4_1_RHEL4_1: 100% [============================]D 0h01m42s
RHEL4_1_RHEL4_1_1: 100% [============================]D 0h00m00s
virt-v2v: RHEL4 configured with virtio drivers.
```

The progress bars are displayed while virt-v2v is copying storage data. In this example, the guest has 2 disks, and progress bars are displayed for both. Note that the storage is copied from the ESX server and written directly to the Red Hat Enterprise Virtualization server in a single step, so the data only has to be 'moved' once.

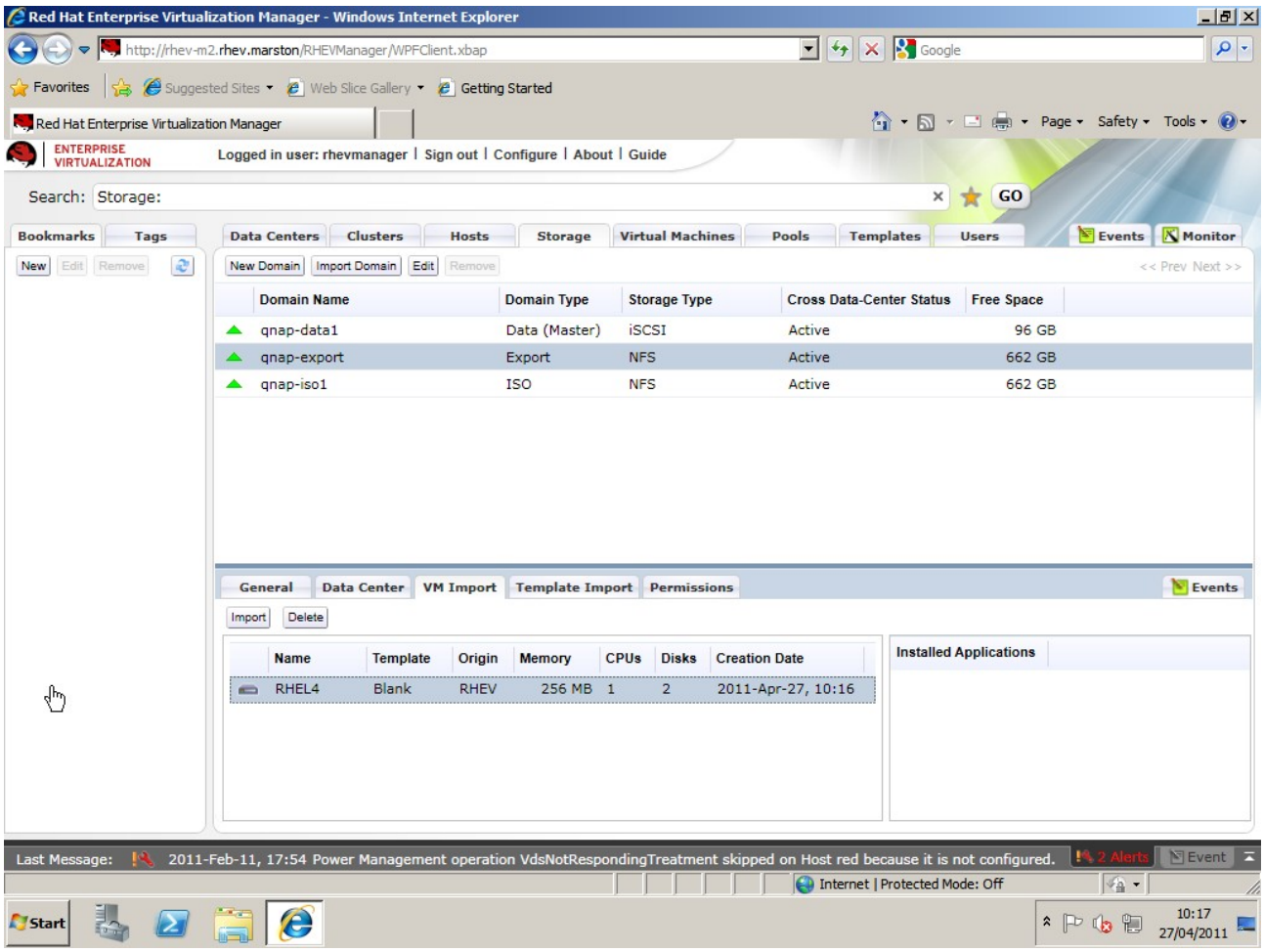
In the above example, the first disk is 2GB, the second only 10MB. Notice that the first disk has taken 1:42 to copy. While this example ran on consumer-grade hardware, it is a useful benchmark. The copy time will grow linearly with disk size, so the time taken for the conversion process to complete is typically dominated by the time taken to copy storage.

The last line indicates that the conversion completed successfully, and that it was possible to enable VirtIO during the process. There are situations where it may not be possible to enable VirtIO, for example because the guest OS doesn't support it, or because the guest required a driver to be installed which wasn't available. The conversion can still succeed in this case, but VirtIO will not be enabled.

Importing into RHEV

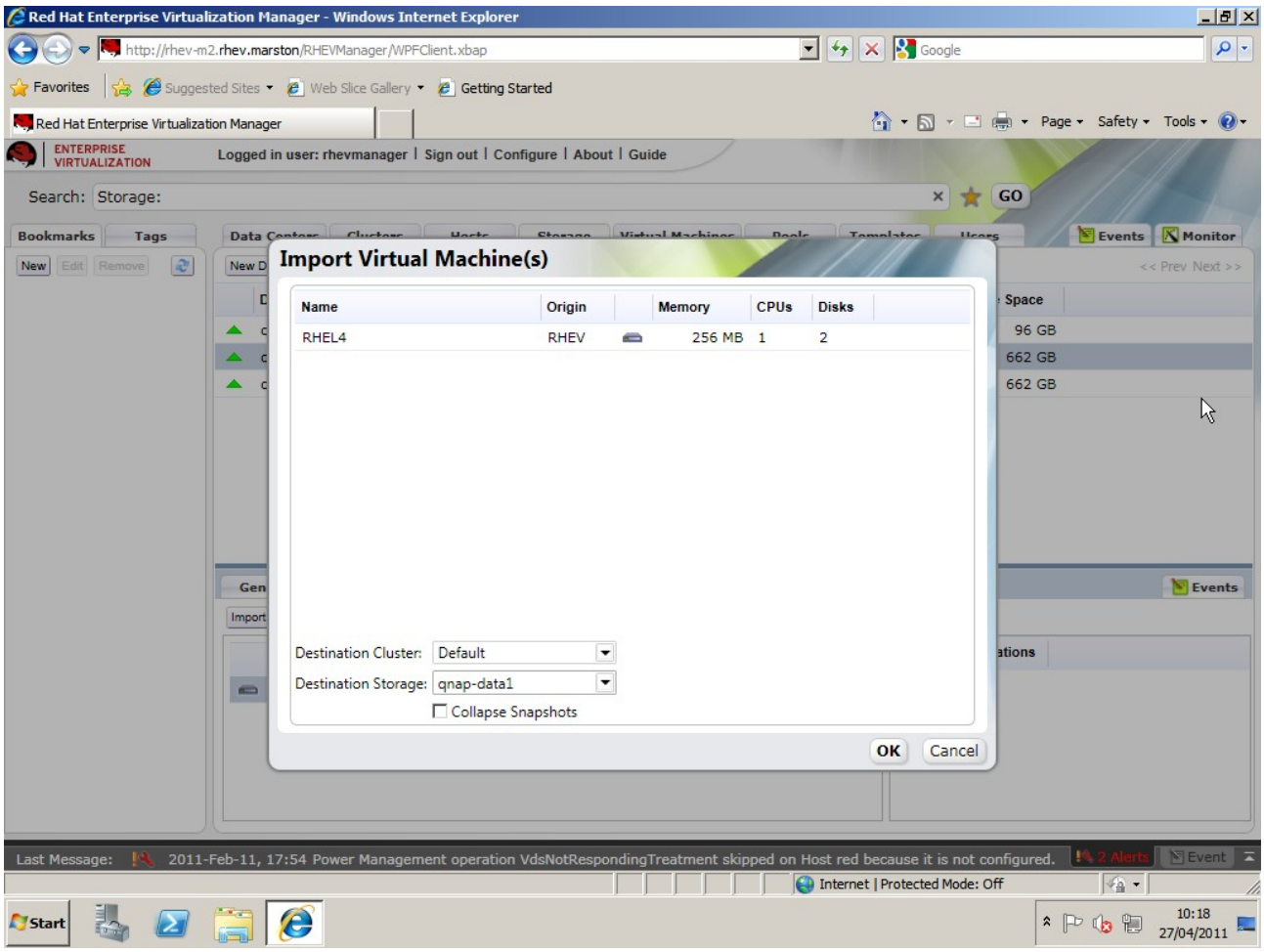
If we were importing to RHEL, the guest would be ready to run at this point. However, when outputting to RHEV we don't directly write a guest which is ready to run. Instead, we write a package in the format RHEV understands to an Export Storage Domain. RHEV can then import this package to a Data Storage Domain, where it can run.

The screenshot below shows the export storage domain in the RHEV admin UI, listing our newly converted guest.

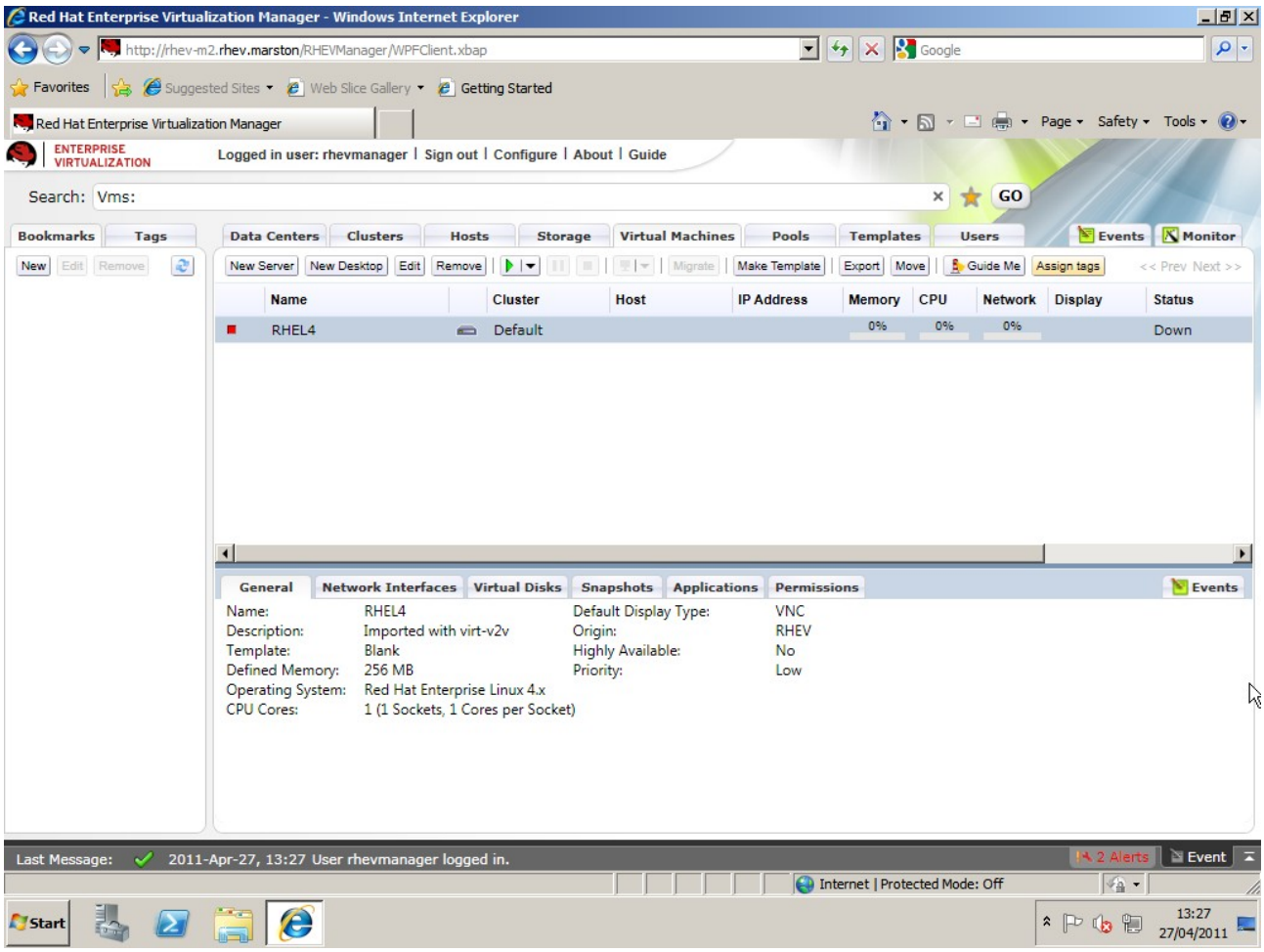


Note that the export storage domain is of type NFS. While RHEV supports export storage domains of type NFS, iSCSI and FCP, virt-v2v is only able to write to an NFS export storage domain. However, this has no bearing on the type of the data storage domain we will import the guest into. Note that in the above example, the data storage domain is of type iSCSI.

On pressing the import button, we see the screen shown below:



On clicking OK, RHEV will start the import process. As this involves copying the guest's storage from the export to the data storage domain, this process can take some time. When it completes, the guest is listed in Virtual Machines tab and is ready to run, as shown below:



The virt-v2v command line

The command line can be broken down into 4 sections:

- *Where to create the target guest*

```
    -o rhev \
    -os qnap.rhev.marston:/rhev-export1 \
```

This specifies that we're outputting to RHEV, and that the target export storage domain has the NFS uri *`qnap.rhev.marston:/rhev-export1`*. If we were outputting to a RHEL target we would specify the name of the destination libvirt storage pool instead.

- *Storage changes*

```
    -of raw -oa preallocated \
```

This specifies that the storage created on RHEV should be raw format, and should be fully preallocated. If necessary, the storage will be converted during the copy. It is possible to convert between qcow2 and raw formats, and preallocated and sparse. Virt-v2v will do zero-detection when converting a preallocated disk image to sparse. Note, however, that format conversion imposes a performance penalty on the copy process.

The data format and allocation policy chosen are especially importing when outputting to RHEV. A RHEV data storage domain, depending on its type can only import certain format/allocation policies. These are shown in the table below (X means supported):

Data Storage Domain Type	raw/sparse	raw/prealloc	qcow2/sparse	qcow2/prealloc
NFS	X	X	X	
iSCSI/FC		X	X	

- *Network changes*

```
    --network rhevm \
```

This specifies that network interfaces should be connected to the network called *rhevm* when imported into RHEV. If the source guest has multiple interfaces which need to be mapped to different virtual networks on the target hypervisor, these can be specified as mappings in virt-v2v.conf. See *virt-v2v.conf(5)* for details.

- *Where the source guest is*

```
    -ic esx://yellow.rhev.marston/?no_verify=1 RHEL4
```

This specifies the location and name of the source guest. As we use libvirt to connect to the ESX server, the source is specified as a libvirt URI. In this case the target is an ESX server called *yellow.rhev.marston*. It also specifies that the server does not have a valid SSL certificate, so libvirt should not attempt to verify it.

There are 2 practical ways to simplify this command line. Firstly, the man page and the V2V guide linked above both contain example command lines for the common scenarios. For the most part, you will be able to copy an example command line and modify the details to suit your local environment.

Secondly, note that the first 3 sections above will be common for a particular target environment. If you are going to be migrating a number of guests to a single target, you can define a profile for that target. In the above case, you would add the following section to /etc/virt-v2v.conf:

```
  <profile name="myrhev">
    <method>rhev</method>
    <storage format="raw" allocation="preallocated">
              qnap.rhev.marston:/rhev-export1
     </storage>
    <network type="default">
      <network type="network" name="rhevm"/>
    </network>
  </profile>
```

Using this newly defined profile, the earlier command line becomes:

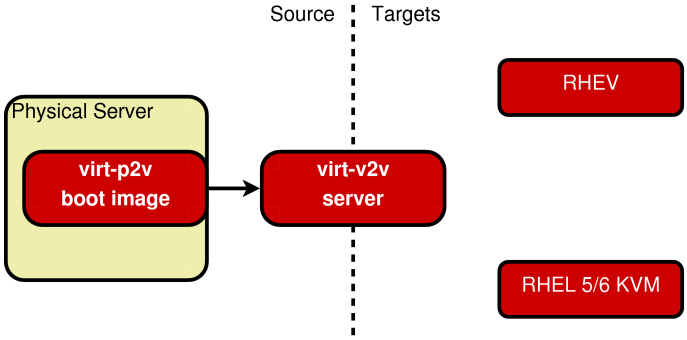
```
# virt-v2v --profile myrhev \
    -ic esx://yellow.rhev.marston/?no_verify=1 RHEL4
```

The P2V tool, introduced below, requires profiles to be defined for all its targets.

Converting workloads from physical servers

While virt-v2v manages the transition between virtualization platforms, it can't access a workload on a physical server. We created the virt-p2v tool to bridge this gap.

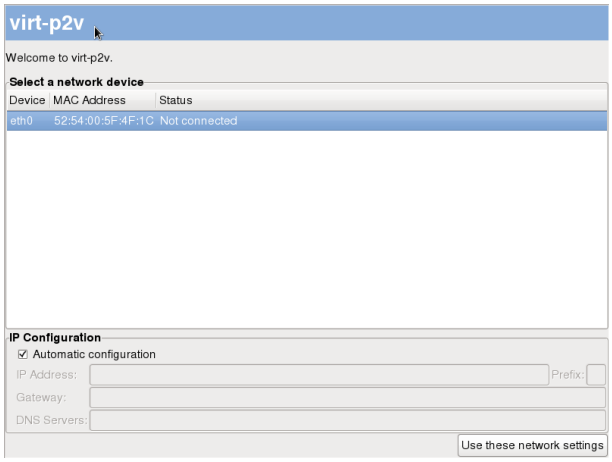
Virt-v2v is able to talk to a hypervisor to obtain guest metadata and storage, but when the source is a physical host there is nothing similar available. To work round this, virt-p2v relies on a small boot image to run the tool. Virt-p2v sends the guest data over SSH to the virt-v2v host, which then converts the guest image as normal and creates the converted guest on the target hypervisor.

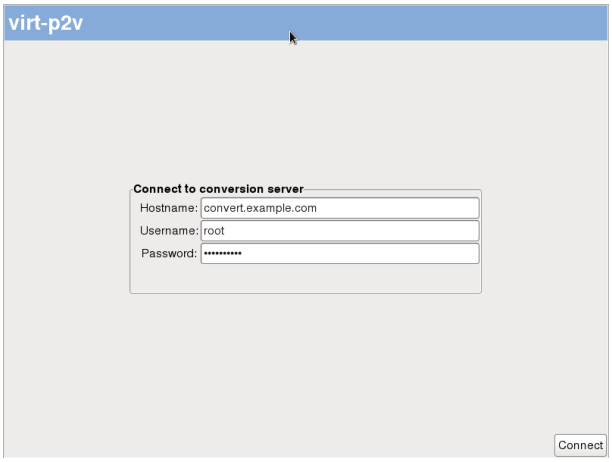


By re-using virt-v2v this way, we can support exactly the same guests to exactly the same targets.

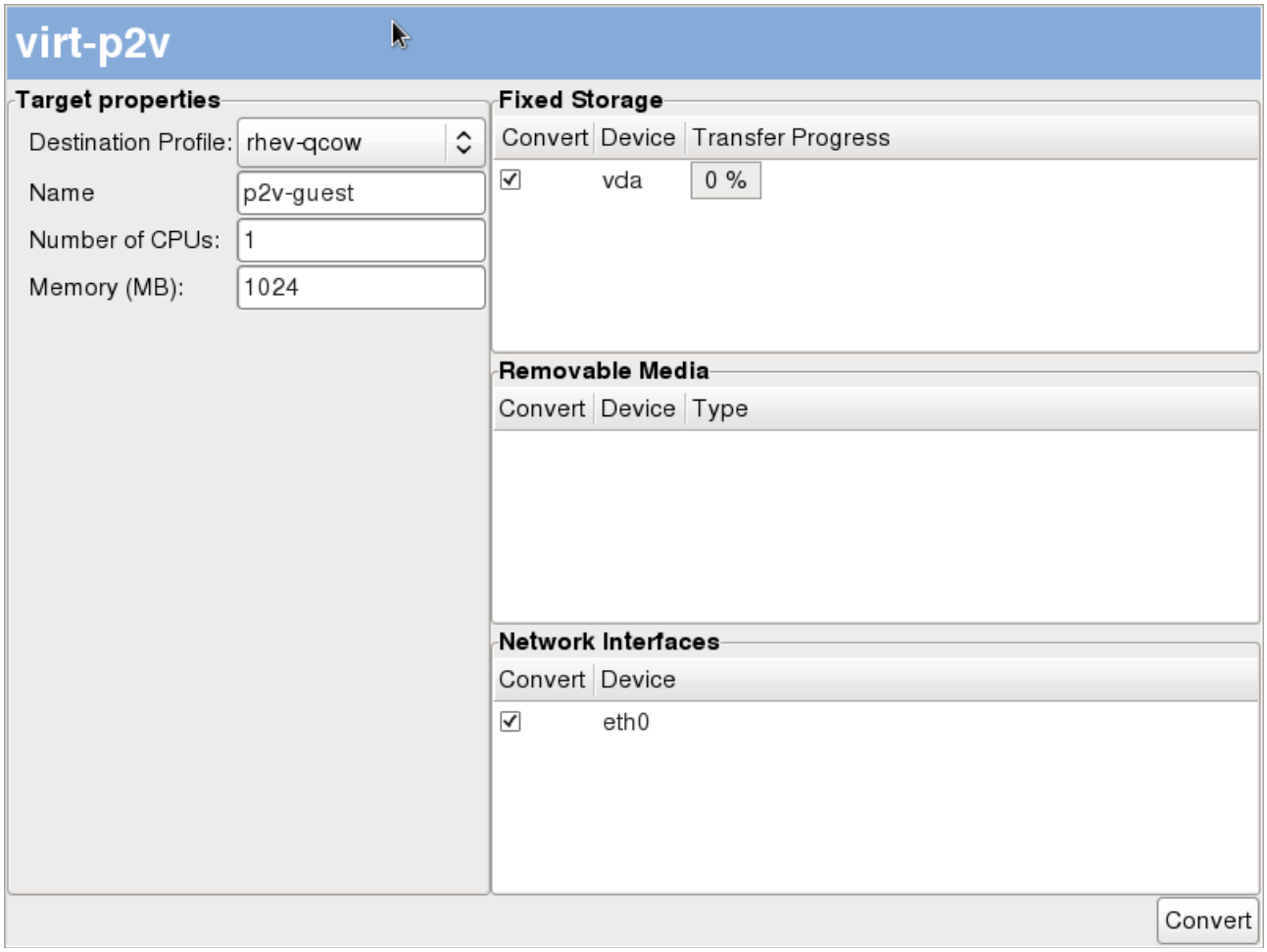
The virt-p2v boot image is created using a separately installable tool, *`virt-p2v-image-builder`*. This tool is a wrapper round the standard Fedora LiveCD tools. As such, the virt-p2v boot image benefits from the various tools available for manipulating Live images. For example it can be written to a USB drive, or converted for PXE delivery. The image comes in under 100MB, making network delivery a practical option.

The P2V process starts with network configuration, followed by connection to a conversion server:





Once connected, the user is prompted with conversion options:



The fields on this page are:

Destination Profile	Where the converted guest will be created. These are profiles defined in /etc/virt-v2v.conf on the conversion server. They define the target server and type, as well as the storage format and allocation policy, and any network mappings. See virt-v2v.conf(5) for more details.
Name	The name the converted guest will be given.
Number of CPUs	The number of CPUs the converted guest will be given. This is automatically populated with the number of cores on the physical machine, but can be edited before conversion.
Memory	The amount of memory the converted guest will be given. Again, this is automatically populated based on the physical machine, and and be edited before conversion.
Fixed Storage	This displays a list of storage devices on the local machine, and allows the user to select which should be transferred during the conversion process.
Removable Media	If the physical host has any floppy or CD-ROM drives, they are listed here. Any selected will also be created on the converted guest.
Network Interfaces	The physical host's network interfaces are listed here. Any selected will be created on the converted guest.

While virt-p2v is intended for use on physical servers, there is no reason that it can't also be used on virtual servers if its workflow better suits your organisation. You may notice that the above screenshots were actually taken on a virtual server.

Status of virt-p2v

Virt-p2v available in Fedora 14, but it is not currently in RHEL. We aim to include it in RHEL in the future. We encourage you to use virt-p2v today, however there are a number of areas we know we have to work on:

- *Transfer speed*

An implementation detail currently means that data transfer is not as fast as it should be. As data transfer dominates the time taken to perform a conversion, it needs to be as fast as possible.

- *Improved fixed storage transfer options*

Modern servers can have an extremely large amount of storage. Often, much of this is unused and unnecessary, and could usefully be excluded from the conversion. This would have the benefits of reducing the conversion time by transferring less data, and using less space on the target environment. We intend to allow the selection of individual partitions on a disk, and resizing of partitions during conversion for supported filesystems.

- *Unattended operation*

It should be possible to specify all options on the boot command line and have the conversion process start immediately. This will make performing multiple, similar conversions simpler and less error-prone.

- *Remove root password requirement for connection to the conversion server*

To improve accountability, it should be possible to specify user, rather than root, credentials when connecting to a conversion server.