XenServer Administrator's Guide

5.0.0
Published September 2008
1.0 Edition
XenServer Administrator's Guide: Release 5.0.0

Published September 2008
Copyright © 2008 Citrix Systems, Inc.

Xen®, Citrix®, XenServer™, XenCenter™ and logos are either registered trademarks or trademarks of Citrix Systems, Inc. in the United States and/or other countries. Other company or product names are for informational purposes only and may be trademarks of their respective owners.

This product contains an embodiment of the following patent pending intellectual property of Citrix Systems, Inc.:


# Table of Contents

1. Overview .................................................................................................................. 1  
   XenServer hosts and resource pools ................................................................. 1  
   Networking ............................................................................................................ 1  
   Storage ................................................................................................................... 1  
   Monitoring and managing XenServer .............................................................. 1  
   Command Line Interface ..................................................................................... 2  
   How this Guide relates to other documentation ................................................. 2  
2. XenServer hosts and resource pools ................................................................. 3  
   Requirements for creating resource pools ......................................................... 3  
   Creating a resource pool ....................................................................................... 4  
   Adding shared storage ........................................................................................... 5  
   Installing and managing VMs on shared storage .............................................. 5  
   Removing a XenServer host from a resource pool ............................................. 6  
   High Availability (HA) ............................................................................................ 7  
      Requirements for configuration ........................................................................ 7  
      Restart priorities ............................................................................................... 8  
   Enabling HA on a XenServer pool ......................................................................... 9  
      Enabling HA using the CLI ............................................................................... 9  
      Removing HA protection from a VM using the CLI ......................................... 9  
      Host Fencing ...................................................................................................... 9  
      Recovering an unreachable host ...................................................................... 10  
      Shutting down a host when HA is enabled ..................................................... 10  
      Shutting down a VM when it is protected by HA ........................................... 10  
   Backups .................................................................................................................. 10  
   Full metadata backup and disaster recovery .................................................... 11  
      Moving SRs between hosts and Pools ................................................................. 13  
   VM Snapshots ........................................................................................................ 16  
      Regular Snapshots ............................................................................................ 16  
      Quiesced Snapshots .......................................................................................... 16  
      Taking a VM snapshot ....................................................................................... 17  
      VM Rollback ...................................................................................................... 17  
   Coping with machine failures .............................................................................. 18  
      Member failures ................................................................................................ 18  
      Master failures .................................................................................................. 18  
      Pool failures ..................................................................................................... 19  
      Coping with Failure due to Configuration Errors .......................................... 19  
      Physical Machine failure .................................................................................. 19  
3. Storage .................................................................................................................... 21  
   Overview ............................................................................................................... 21  
      Storage repositories (SRs) ............................................................................... 21  
      Virtual Disk Images (VDI) ................................................................................ 21  
      Managing Storage .............................................................................................. 22  
   Storage Repository Types .................................................................................... 23  
      Local Disks ......................................................................................................... 24  
      Local hotplug devices ....................................................................................... 24  
      Shared Network Attached Storage using NFS ............................................... 24  
      Shared iSCSI Storage ......................................................................................... 25  
      Shared LVM storage over FC or iSCSI hardware HBAs ................................... 26  
      Shared NetApp Storage ..................................................................................... 27  
      Shared EqualLogic Storage ............................................................................. 29  
   Storage configuration examples .......................................................................... 29
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. Networking</td>
<td>46</td>
</tr>
<tr>
<td>Creating Storage Repositories</td>
<td>30</td>
</tr>
<tr>
<td>Probing an SR</td>
<td>30</td>
</tr>
<tr>
<td>Creating a local LVM SR (lvm)</td>
<td>33</td>
</tr>
<tr>
<td>Creating a local EXT3 SR (ext)</td>
<td>33</td>
</tr>
<tr>
<td>Creating a shared NFS SR (nfs)</td>
<td>34</td>
</tr>
<tr>
<td>Creating a shared LVM over iSCSI SR using the software iSCSI initiator (lvmiscsi)</td>
<td>34</td>
</tr>
<tr>
<td>Creating a shared LVM over Fibre Channel or iSCSI HBA SR (lvmohba)</td>
<td>34</td>
</tr>
<tr>
<td>Creating a shared NetApp SR over iSCSI</td>
<td>37</td>
</tr>
<tr>
<td>Creating a shared EqualLogic SR</td>
<td>38</td>
</tr>
<tr>
<td>Storage Multipathing</td>
<td>39</td>
</tr>
<tr>
<td>Managing Storage Repositories</td>
<td>40</td>
</tr>
<tr>
<td>Destroying or forgetting a SR</td>
<td>40</td>
</tr>
<tr>
<td>Introducing an SR</td>
<td>40</td>
</tr>
<tr>
<td>Converting local Fibre Channel SRs to shared SRs</td>
<td>41</td>
</tr>
<tr>
<td>Moving Virtual Disk Images (VDIs) between SRs</td>
<td>41</td>
</tr>
<tr>
<td>Managing VDIs in a NetApp SR</td>
<td>42</td>
</tr>
<tr>
<td>Taking VDI snapshots with a NetApp SR</td>
<td>43</td>
</tr>
<tr>
<td>Adjusting the disk IO scheduler for an LVM-based SR</td>
<td>43</td>
</tr>
<tr>
<td>Managing Host Bus Adapters (HBAs)</td>
<td>43</td>
</tr>
<tr>
<td>Sample QLogic iSCSI HBA setup</td>
<td>43</td>
</tr>
<tr>
<td>Removing HBA-based FC or iSCSI device entries</td>
<td>44</td>
</tr>
<tr>
<td>Enabling Virtual HBA support</td>
<td>44</td>
</tr>
<tr>
<td>Virtual disk QoS settings (Enterprise Edition only)</td>
<td>44</td>
</tr>
<tr>
<td>XenServer networking concepts</td>
<td>46</td>
</tr>
<tr>
<td>Network objects</td>
<td>46</td>
</tr>
<tr>
<td>Networks</td>
<td>46</td>
</tr>
<tr>
<td>VLANs</td>
<td>47</td>
</tr>
<tr>
<td>NIC bonds</td>
<td>47</td>
</tr>
<tr>
<td>Initial networking configuration</td>
<td>48</td>
</tr>
<tr>
<td>Managing networking configuration</td>
<td>49</td>
</tr>
<tr>
<td>Creating networks in a standalone server</td>
<td>49</td>
</tr>
<tr>
<td>Creating networks in resource pools</td>
<td>49</td>
</tr>
<tr>
<td>Creating VLANs</td>
<td>49</td>
</tr>
<tr>
<td>Creating NIC bonds on a standalone host</td>
<td>50</td>
</tr>
<tr>
<td>Creating NIC bonds in resource pools</td>
<td>51</td>
</tr>
<tr>
<td>Configuring a dedicated storage NIC</td>
<td>55</td>
</tr>
<tr>
<td>Controlling Quality of Service (QoS)</td>
<td>56</td>
</tr>
<tr>
<td>Changing networking configuration options</td>
<td>56</td>
</tr>
<tr>
<td>NIC/PIF ordering in resource pools</td>
<td>58</td>
</tr>
<tr>
<td>Networking Troubleshooting</td>
<td>60</td>
</tr>
<tr>
<td>Diagnosing network corruption</td>
<td>60</td>
</tr>
<tr>
<td>Recovering from a bad network configuration</td>
<td>60</td>
</tr>
<tr>
<td>5. Monitoring and managing XenServer</td>
<td>62</td>
</tr>
<tr>
<td>Alerts</td>
<td>62</td>
</tr>
<tr>
<td>Customizing Alerts</td>
<td>63</td>
</tr>
<tr>
<td>Configuring Email Alerts</td>
<td>64</td>
</tr>
<tr>
<td>Custom Fields and Tags</td>
<td>64</td>
</tr>
<tr>
<td>Custom Searches</td>
<td>65</td>
</tr>
<tr>
<td>6. Command line interface</td>
<td>66</td>
</tr>
<tr>
<td>Basic xe syntax</td>
<td>66</td>
</tr>
<tr>
<td>Special characters and syntax</td>
<td>67</td>
</tr>
<tr>
<td>Command types</td>
<td>68</td>
</tr>
</tbody>
</table>
Parameter types ................................................................. 69
Low-level param commands ................................................ 69
Low-level list commands ..................................................... 70
\textbf{x}e command reference ................................................ 71
Bonding commands ............................................................. 71
CD commands ................................................................. 71
Console commands ............................................................ 73
Event commands ............................................................... 74
Host (XenServer host) commands ......................................... 75
Log commands ................................................................. 83
Network commands ........................................................... 84
Patch (update) commands ................................................... 85
PBD commands ................................................................. 86
PIF commands ................................................................. 87
Pool commands ............................................................... 90
Storage Manager commands ............................................... 93
SR commands ................................................................. 94
Task commands ............................................................... 96
Template commands .......................................................... 97
Update commands ............................................................. 103
User commands ............................................................... 103
VBD commands ............................................................... 103
VDI commands ............................................................... 105
VIF commands ............................................................... 109
VLAN commands ............................................................. 111
VM commands ............................................................... 111
7. Troubleshooting ............................................................... 124
XenServer host logs ............................................................ 124
Sending log messages to a central server ................................ 125
Troubleshooting connections between XenCenter and the XenServer host ........................................... 125
A. CPU Allocation Guidelines ............................................... 126
Index .................................................................................. 127
Chapter 1. Overview

This document is a system administrator's guide to XenServer™, the platform virtualization solution from Citrix™. It describes the tasks involved in configuring a XenServer deployment -- in particular, how to set up storage, networking and resource pools, and how to administer XenServer hosts using the xe command line interface (CLI).

This section summarizes the rest of the guide so that you can find the information you need. The following topics are covered:

• XenServer hosts and resource pools
• XenServer network configuration
• XenServer storage configuration
• Monitoring and Managing XenServer
• XenServer command line interface

XenServer hosts and resource pools

A resource pool is a connected group of up to 16 XenServer hosts that, combined with shared storage, provides a platform on which VMs run. VMs can be started on different hosts within the resource pool, and can even be live migrated between pool hosts in order to minimize downtime.

The XenServer hosts and resource pools chapter introduces the concept of resource pools and describes how to:

• add and remove XenServer hosts from pools
• create shared storage and attach it to a pool
• start VMs on different XenServer hosts within a pool
• live migrate running VMs between XenServer hosts within a pool

Networking

The Networking chapter introduces physical and virtual network concepts, describing how to:

• configure physical networking on XenServer hosts and resource pools
• create virtual network interfaces for VMs, and bridge these to physical networks
• work with VLANs

Storage

The Storage chapter introduces physical and virtual storage concepts, describing how to:

• create shared and local storage repositories on a variety of different substrates (including iSCSI, NFS and Fibre Channel)
• create virtual disk images within storage repositories as part of the process of installing VMs
• manage and administer storage repositories

Monitoring and managing XenServer

The Monitoring and managing XenServer chapter explains how to configure events and alerts. The Monitoring guide describes:
• XenServer events and alerts

**Command Line Interface**

The Command Line Interface chapter introduces "xe": a powerful CLI that facilitates administration of all aspects of XenServer, including host configuration, storage, networking and VMs. The CLI guide describes:

• the syntax of xe commands
• using xe in both on- and off-host modes on both Windows and Linux
• using xe to query the parameters of physical and virtual objects, including hosts, networks, storage repositories, virtual disks and VMs
• using xe to configure XenServer deployments (including host, network, storage and VM configuration)

**How this Guide relates to other documentation**

This document is primarily aimed at system administrators, who need to configure and administer XenServer deployments. Other documentation shipped with this release includes:

• *XenServer Installation Guide* provides a high level overview of XenServer, along with step-by-step instructions on installing XenServer hosts and the XenCenter management console.

• *XenServer Virtual Machine Installation Guide* describes how to install Linux and Windows VMs on top of a XenServer deployment. As well as installing new VMs from install media (or via the VM templates provided with the XenServer release), this guide also explains how to create VMs from existing physical machines, via a process called P2V.

• *XenServer Software Development Kit Guide* presents an overview of the XenServer SDK -- a selection of code samples that demonstrate how to write applications that interface with XenServer hosts.

• *XenAPI Specification* provides a programmer's reference guide to the XenServer API.

• *Release Notes* provides a list of known issues that affect this release.
Chapter 2. XenServer hosts and resource pools

A resource pool comprises multiple XenServer host installations, bound together into a single managed entity which can host Virtual Machines. When combined with shared storage, a resource pool enables VMs to be started on any XenServer host which has sufficient memory and then dynamically moved between XenServer hosts while running with minimal downtime (XenMotion). If an individual XenServer host suffers a hardware failure, then the administrator can restart the failed VMs on another XenServer host in the same resource pool. If high availability (HA) is enabled on the resource pool, VMs will automatically be moved if their host fails. Up to 16 hosts are supported per resource pool, although this restriction is not enforced.

This chapter describes how resource pools can be created through a series of examples using the xe command line interface (CLI). A simple NFS-based shared storage configuration is presented and a number of simple VM management examples are discussed. Procedures for dealing with physical node failures are also described.

A pool always has at least one physical node, known as the master. Other physical nodes join existing pools and are described as members. Only the master node exposes an administration interface (used by XenCenter and the CLI); the master will forward commands to individual members as necessary.

Requirements for creating resource pools

A resource pool is an aggregate of one or more homogeneous XenServer hosts, up to a maximum of 16. The definition of homogeneous is:

- each CPU is from the same vendor (in particular AMD-V and Intel VT CPUs cannot be mixed)
- each CPU is the same model (except for stepping)
- each CPU has the same feature flags
- all hosts are running the same version of XenServer software

**Note**

This restriction is relaxed during rolling upgrade.

In addition to being homogeneous, an individual XenServer host can only join a resource pool if:

- an Enterprise license is installed on the host
- it has a static IP address (either manually assigned or via DHCP);
- it is not a member of an existing resource pool
- its clock is synchronized to the same time source as the pool master (for example, via NTP)
- it has no shared storage configured
- there are no running or suspended VMs on the XenServer host which is joining
- there are no active operations on the VMs in progress, such as one shutting down
- the management NIC of the XenServer host which is joining is not part of a NIC bond

XenServer hosts in resource pools may contain different numbers of physical network interfaces. Local storage repositories may also exist, of varying size. In practice, it is often difficult to obtain multiple servers with the exact same CPUs, and so minor variations are permitted. If you are sure that it is acceptable in your environment for hosts with varying CPUs to be part of the same resource pool, then the pool joining operation can be forced.
Although not a strict technical requirement for creating a resource pool, the advantages of pools (for example, the ability to dynamically choose on which XenServer host to run a VM and to dynamically move a VM between XenServer hosts) are only available if the pool has one or more shared storage repositories. As a rule of thumb, you should postpone creating a pool of XenServer hosts until shared storage is available. Once shared storage has been added, we recommend you to move existing VMs whose disks are in local storage into shared storage. This can be done using the \texttt{xe vm-copy} command or XenCenter.

### Creating a resource pool

Resource pools can be created using either the XenCenter management console or the CLI. When you join a new host to a resource pool, the joining host synchronizes its local database with the pool-wide one, and inherits some settings from the pool:

- VMs running on local storage are added to the pool-wide database, and local storage (and non-shared remote storage) are also merged. All of these will still be tied to the joining host in the pool unless you explicitly take action to make the resources shared after the join has completed.

- The joining host inherits existing shared storage repositories in the pool and appropriate PBD records are created so that the new host can access existing shared storage automatically.

- Networking information is partially inherited to the joining host: the \textit{structural} details of NICs, VLANs and bonded interfaces are all inherited, but \textit{policy} information is not. This policy information which must be re-configured includes:
  - the IP addresses of management NICs, which are preserved from the original configuration
  - the location of the management interface remains the same as the original configuration. For example, if the other pool hosts have their management interface on a bonded interface, then the joining host must be explicitly migrated to the bond once it has joined. See \textbf{To add NIC bonds to existing pool master and member hosts} for details on how to migrate the management interface to a bond.
  - Dedicated storage NICs must be re-assigned to the joining host from XenCenter or the CLI, and the PBDs re-plugged to route the traffic accordingly. This is because IP addresses are not assigned as part of the pool join operation, and the storage NIC is not useful without this configured correctly. See \textbf{the section called “Configuring a dedicated storage NIC”} for details on how to dedicate a storage NIC from the CLI.

### To join XenServer hosts \texttt{host1} and \texttt{host2} into a resource pool via the CLI

1. Open a console on XenServer host \texttt{host2}.
2. Command XenServer host \texttt{host2} to join the pool on XenServer host \texttt{host1} by issuing the command:

   ```
   xe pool-join master-address=<host1> master-username=<root> \ master-password=<password>
   ```

   The \texttt{master-address} must be set to the fully-qualified domain name of XenServer host \texttt{host1} and the \texttt{password} must be the administrator password set when XenServer host \texttt{host1} was installed.

###Naming a resource pool

- XenServer hosts belong to an unnamed pool by default. To create a resource pool, rename the existing nameless pool:
Adding shared storage

For a complete list of supported shared storage types, see the Storage chapter. This section demonstrates how shared storage (represented as a storage repository) can be created on an existing NFS server.

Adding NFS shared storage to a resource pool via the CLI

1. Open a text console on any XenServer host in the pool.
2. Create the storage repository on `<server:/path>` by issuing the command

   ```
   xe sr-create content-type=user type=nfs name-label="Example SR" shared=true \
   device-config:server=<server> \
   device-config:serverpath=<path>
   ```

   The `device-config:server` refers to the hostname of the NFS server and `device-config:serverpath` refers to the path on the NFS server. Since `shared` is set to true, the shared storage will be automatically connected to every XenServer host in the pool and any XenServer hosts that subsequently join will also be connected to the storage. The UUID of the created storage repository will be printed on the screen.

3. Find the UUID of the pool by the command

4. Set the shared storage as the pool-wide default with the command

   ```
   xe pool-param-set uuid=<pool-uuid> default-SR=<sr-uuid>
   ```

   Since the shared storage has been set as the pool-wide default, all future VMs will have their disks created on shared storage by default. See Chapter 3, Storage for information about creating other types of shared storage.

Installing and managing VMs on shared storage

The following example shows how to install a Debian Linux VM using the Debian Etch 4.0 template provided with XenServer.

Installing a Debian Etch (4.0) VM

1. Open a console on any host in the pool.
2. Use the `sr-list` command to find the UUID of your shared storage:

   ```
   xe sr-list
   ```

3. Create the Debian VM by issuing the command

   ```
   xe vm-install template="Debian Etch 4.0" new-name-label=<etch> \
   sr_uuid=<shared_storage_uuid>
   ```

   When the command completes, the Debian VM will be ready to start.

4. Start the Debian VM with the command

   ```
   xe vm-start vm=etch
   ```
The master will choose a XenServer host from the pool to start the VM. If the on parameter is provided, the VM will start on the specified XenServer host. If the requested XenServer host is unable to start the VM, the command will fail. To request that a VM is always started on a particular XenServer host, set the affinity parameter of the VM to the UUID of the desired XenServer host using the \texttt{xe vm-param-set} command. Once set, the system will start the VM there if it can; if it cannot, it will default to choosing from the set of possible XenServer hosts.

5. You can use XenMotion to move the Debian VM to another XenServer host with the command

\texttt{xen}\_\texttt{vm-migrate vm=<etch> host=<host\_name> --live}

XenMotion keeps the VM running during this process to minimize downtime.

\begin{quote}
\textbf{Note}

When a VM is migrated, the domain on the original hosting server is destroyed and the memory that VM used is zeroed out before Xen makes it available to new VMs. This ensures that there is no information leak from old VMs to new ones. As a consequence, it is possible that sending multiple near-simultaneous commands to migrate a number of VMs, when near the memory limit of a server (for example, a set of VMs consuming 3GB migrated to a server with 4GB of physical memory), the memory of an old domain might not be scrubbed before a migration is attempted, causing the migration to fail with a HOST\_NOT\_ENOUGH\_FREE\_MEMORY error. Inserting a delay between migrations should allow Xen the opportunity to successfully scrub the memory and return it to general use.
\end{quote}

\section*{Removing a XenServer host from a resource pool}

When a XenServer host is removed (ejected) from a pool, the machine is rebooted, reinitialized, and left in a state equivalent to that after a fresh installation. It is important not to eject a XenServer host from a pool if there is important data on the local disks.

\textbf{To remove a host} \textit{b} \textbf{from a resource pool using the CLI}

1. Open a console on any host in the pool.
2. Find the UUID of the host \textit{b} using the command

\texttt{xen}\_\texttt{host-list}

3. Eject the host from the pool:

\texttt{xen}\_\texttt{pool-eject host-uuid=<uuid>}

The XenServer host will be ejected and left in a freshly-installed state.

\begin{quote}
\textbf{Warning}

Do not eject a host from a resource pool if it contains important data stored on its local disks. All of the data will be erased upon ejection from the pool. If you wish to preserve this data, copy the VM to shared storage on the pool first using XenCenter, or the \texttt{xen}\_\texttt{vm-copy} CLI command.
\end{quote}

When a member XenServer host containing locally stored VMs is ejected from a pool, those VMs will still be present in the pool database and visible to the other XenServer hosts. They will not start until the virtual disks associated with
them have been changed to point at shared storage which can be seen by other XenServer hosts in the pool, or simply removed. It is for this reason that you are strongly advised to move any local storage to shared storage upon joining a pool, so that individual XenServer hosts can be ejected (or physically fail) without loss of data.

**High Availability (HA)**

When the HA feature is enabled, XenServer continually monitors the health of the hosts in a pool. The HA mechanism automatically moves protected VMs to a healthy host if the current VM host fails. Additionally, if the host that fails is the master, HA selects another host to take over the master role automatically, meaning that you will be able to continue to manage the XenServer pool.

In order to absolutely guarantee that a host is unreachable, a resource pool configured for high-availability uses several 'heartbeat' mechanisms to regularly check up on hosts. These heartbeats go through both the storage interfaces (via the 'Heartbeat SR') and also the networking interfaces (via the management interfaces). Both of these heartbeat routes can be multi-homed for additional resilience to prevent false positives from component failures.

XenServer dynamically maintains a 'failover plan' for what to do if a set of hosts in a pool fail at any given time. An important concept to understand is the 'host failures to tolerate' value, which is defined as part of HA configuration. This determines the number of failures which is allowed without any loss of service. For example, if a resource pool consisted of 16 hosts, and the tolerated failures is set to 3, then the pool will calculate a failover plan which allows for any 3 hosts to fail and still be able to restart VMs on other hosts. If a plan cannot be found, then the pool is considered to be 'overcommitted'. The plan is dynamically recalculated based on VM lifecycle operations and movement. Alerts are sent (either via XenCenter or e-mail) if changes (for example the addition on new VMs to the pool) cause your pool to become overcommitted.

**Requirements for configuration**

In order to use the HA feature, you will require:

- shared storage, including at least one iSCSI or Fibre Channel LUN of size 356MiB or greater -- the 'heartbeat SR'.
  The HA mechanism creates two volumes on the heartbeat SR:
  - 4MiB heartbeat volume
    Used for heartbeating.
  - 256MiB metadata volume
    Stores pool master metadata to be used in the case of master failover.

  If you are using a NetApp or EqualLogic SR, then you should manually provision an iSCSI LUN on the array to use for the heartbeat SR.

- a XenServer pool (this feature provides high availability at the server level within a single resource pool).

- Enterprise licenses on all hosts.

In order for a virtual machine to be protected by the HA feature, it must be agile. This means that it must have its virtual disks on shared storage (any type of shared storage may be used; the iSCSI or Fibre Channel LUN is only required for the storage heartbeat and can be used for virtual disk storage if you prefer, but this is not necessary), must not have a connection to a local DVD drive configured, and should have its virtual network interfaces on pool-wide networks.

We strongly recommend the use of a bonded management interface on the servers in the pool if HA is enabled, and multipathed storage for the Heartbeat SR.

If you create VLANs and bonded interfaces from the CLI, then they may not be plugged in and active despite being created. In this situation, a VM can appear to be not agile, and cannot be protected by HA. If this occurs, use the CLI `pif-plug` command to bring the VLAN and bond PIFs up so that the VM can become agile. You can also determine precisely why a VM is not agile by using the `xe diagnostic-vm-status` CLI command to analyze its placement constraints, and take remedial action if required.
Restart priorities

Virtual machines are assigned a restart priority and a flag that indicates whether they should be protected by HA or not. When HA is enabled, every effort is made to keep protected virtual machines live. If a restart priority is specified, any protected VM that is halted will be started automatically. If a server fails then the VMs on it will be started on another server.

The possible restart priorities are:

1 | 2 | 3

when a pool is overcommitted the HA mechanism will attempt to restart protected VMs with the lowest restart priority first

best-effort

VMs with this priority setting will be restarted only when the system has attempted to restart protected VMs

ha-always-run=false

VMs with this parameter set will not be restarted

The restart priorities determine the order in which VMs are restarted when a failure occurs. In a given configuration where a number of server failures greater than zero can be tolerated (as indicated in the HA panel in the GUI, or by the ha-plan-exists-for field on the pool object in the CLI), the VMs that have restart priorities 1, 2 or 3 are guaranteed to be restarted given the stated number of server failures. VMs with a best-effort priority setting are not part of the failover plan and are not guaranteed to be kept running, since capacity is not reserved for them. If the pool experiences server failures and enters a state where the number of tolerable failures drops to zero, the protected VMs will no longer be guaranteed to be restarted. If this condition is reached, a system alert will be generated. In this case, should an additional failure occur, all VMs that have a restart priority set will behave according to the best-effort behavior.

If a protected VM cannot be restarted at the time of a server failure (for example, if the pool was overcommitted when the failure occurred), further attempts to start this VM will be made as the state of the pool changes. This means that if extra capacity becomes available in a pool (if you shut down a non-essential VM, or add an additional server, for example), a fresh attempt to restart the protected VMs will be made, which may now succeed.

**Note**

No running VM will ever be stopped or migrated in order to free resources for a VM with always-run=true to be restarted.

Overcommitting

A pool is overcommitted if the VMs that are currently running could not be restarted elsewhere following a user-defined number of failures.

This would happen if there was not enough free memory across the pool to run those VMs following failure. However there are also more subtle changes which can make HA guarantees unsustainable: changes to VBDs and networks can affect which VMs may be restarted on which hosts. Currently it is not possible for XenServer to check all actions before they occur and determine if they will cause violation of HA demands. However an asynchronous notification is sent if HA becomes unsustainable.

Overcommitment Warning

If you attempt to start or resume a VM and that action causes the pool to be overcommitted, a warning alert is raised. This warning is displayed in XenCenter and is also available as a message instance through the Xen API. The message may also be sent to an email address if configured. You will then be allowed to cancel the operation, or proceed anyway.
This will cause the pool to become overcommitted. The amount of memory used by VMs of different priorities is displayed at the pool and host levels.

## Enabling HA on a XenServer pool

HA can be enabled on a pool using either XenCenter or the command-line interface. In either case, you will specify a set of priorities that determine which VMs should be given highest restart priority when a pool is overcommitted.

### Warning

When HA is enabled, some operations that would compromise the plan for restarting VMs may be disabled, such as removing a server from a pool. To perform these operations, HA can be temporarily disabled, or alternately, VMs protected by HA made unprotected.

## Enabling HA using the CLI

1. Verify that you have a compatible Storage Repository (SR) attached to your pool. iSCSI or Fibre Channel are compatible SR types. Please refer to the reference guide for details on how to configure such a storage repository using the CLI.

2. For each VM you wish to protect, set a restart priority. You can do this as follows:

   ```
   xe vm-param-set uuid=<vm_uuid> ha-restart-priority=<1> ha-always-run=true
   ```

3. Enable HA on the pool:

   ```
   xe pool-ha-enable heartbeat-sr-uuid=<sr_uuid>
   ```

4. Run the `pool-ha-compute-max-host-failures-to-tolerate` command. This command returns the maximum number of hosts that can fail before there are insufficient resources to run all the protected VMs in the pool.

   ```
   xe pool-ha-compute-max-host-failures-to-tolerate
   ```

   The number of failures to tolerate determines when an alert is sent: the system will recompute a failover plan as the state of the pool changes and with this computation the system identifies the capacity of the pool and how many more failures are possible without loss of the liveness guarantee for protected VMs. A system alert is generated when this computed value falls below the specified value for `ha-host-failures-to-tolerate`.

5. Specify the number of failures to tolerate parameter. This should be less than or equal to the computed value:

   ```
   xe pool-param-set ha-host-failures-to-tolerate=<2>
   ```

## Removing HA protection from a VM using the CLI

To disable HA features for a VM, use the `xe vm-param-set` command to set the `ha-always-run` parameter to `false`. This will not clear the VM restart priority settings. You can enable HA for a VM again by setting the `ha-always-run` parameter to `true`.

## Host Fencing

If a server failure occurs such as the loss of network connectivity or a problem with the control stack is encountered, the XenServer host will self-fence to ensure that the virtual machines are not running on two servers simultaneously. When a fence action is taken, the server will immediately and abruptly be restarted, causing all virtual machines running...
on it to be stopped. The other servers will detect that the virtual machines are no longer running and they will be restart according to the restart priorities assign to them. The fenced server will enter a reboot sequence, and when it has restarted it will attempt to re-join the resource pool.

**Recovering an unreachable host**

If for some reason a host cannot access the HA statefile, it is possible that a host may become unreachable. To recover your XenServer installation it may be necessary to disable HA using the `host-emergency-ha-disable` command:

```
xe host-emergency-ha-disable --force
```

If the host was the Pool master, then it should start up as normal with HA disabled. Slaves should reconnect and automatically disable HA. If the host was a Pool slave and cannot contact the master, then it may be necessary to force the host to reboot as a pool master (`xe pool-emergency-transition-to-master`) or to tell it where the new master is (`xe pool-emergency-reset-master`):

```
xe pool-emergency-transition-to-master uuid=<host_uuid>
xe pool-emergency-reset-master master-address=<new_master_hostname>
```

When all hosts have successfully restarted, re-enable HA:

```
xe pool-ha-enable heartbeat-sr-uuid=<sr_uuid>
```

**Shutting down a host when HA is enabled**

When HA is enabled special care needs to be taken when shutting down or rebooting a host to prevent the HA mechanism from assuming that the host has failed. To shut down a host cleanly in an HA-enabled environment, first disable the host, then evacuate the host and finally shutdown the host using either XenCenter or the CLI. To shutdown a host in an HA-enabled environment on the command line:

```
xe host-disable host=<host_name>
xe host-evacuate uuid=<host_uuid>
xe host-shutdown host=<host_name>
```

**Shutting down a VM when it is protected by HA**

When a VM is protected under a HA plan and set to automatically restart, it cannot be shut down while this protection is active. In order to shut down a VM, first disable its HA protection and then execute the CLI command. XenCenter offers you a dialog box to automate disabling the protection if you click on the Shutdown button of a protected VM.

Note that if you shut down a VM from within the guest, and the VM is protected, then it will automatically be restarted under the HA failure conditions. This helps ensure that operator error (or an errant program which mistakenly shuts down the VM) does not result in a protected VM being left shut down accidentally. If you do wish to shut this VM down, then disable its HA protection first.

**Backups**

Citrix recommends that you frequently perform as many of the following backup procedures as possible to recover from possible server and/or software failure.

**To backup pool metadata**

1. Run the command:

```
xe pool-dump-database file-name=<backup>
```
2. Run the command:

```
xe pool-restore-database file-name=<backup> --dry-run
```

This command checks that the target machine has an appropriate number of appropriately named NICs, which is required for the backup to succeed.

**To backup host configuration and software**

- Run the command:

```
xe host-backup host=<host> file-name=<hostbackup>
```

**Note**
- Do not create the backup in domain 0.
- This procedure may create a large backup file.
- To complete a restore you have to reboot to the original install CD.
- This data can only be restored to the original machine.

**To backup a VM**

1. Ensure that the VM to be backed up is offline.
2. Run the command:

```
xe vm-export vm=<vm_uuid> filename=<backup>
```

**Note**
This backup also backs up all of the VM's data. When importing a VM, you can specify the storage mechanism to use for the backed up data.

**Warning**
Because this process backs up all of the VM data, it can take some time to complete.

**To backup VM metadata only**

- Run the command:

```
xe vm-export vm=<vm_uuid> filename=<backup> --metadata
```

**Full metadata backup and disaster recovery**

XenServer 5.0.0 introduces the concept of Portable Storage Repositories (Portable SRs). Portable SRs contain all of the information necessary to recreate all the Virtual Machines (VMs) with Virtual Disk Images (VDIs) stored on the
SR after re-attaching the SR to a different host or pool. Portable SRs can be used when regular maintenance or disaster recovery requires manually moving a SR between pools or standalone hosts.

Using portable SRs has similar constraints to XenMotion as both cases result in VMs being moved between hosts. To use portable SRs:

- The source and destination hosts must have the same CPU type (ie Intel or AMD) and networking configuration
- The SR media itself, such as a LUN for iSCSI and FibreChannel SRs, must be able to be moved, re-mapped, or replicated between the source and destination hosts
- If using tiered storage, where a VM has VDIs on multiple SRs, all required SRs must be moved to the destination host or pool
- Any configuration data required to connect the SR on the destination host or pool, such as the target IP address, target IQN, and LUN SCSI ID for iSCSI SRs, and the LUN SCSI ID for FibreChannel SRs, must be maintained manually
- The backup metadata option must be configured for the desired SR

**Note**

When moving portable SRs between pools the source and destination pools are not required to have the same number of hosts. Moving portable SRs between pools and standalone hosts is also supported provided the above constraints are met.

Portable SRs work by creating a dedicated metadata VDI within the specified SR. The metadata VDI is used to store copies of the pool or host database as well as the metadata describing each VM's configuration. As a result the SR becomes fully self-contained, or portable, allowing it to be detached from one host and re-attached to another as a new SR. Once the SR is re-attached a restore process is used to recreate all of the VMs on the SR from the metadata VDI. For disaster recovery the metadata backup can be scheduled to run regularly to ensure the metadata SR is current.

The metadata backup and restore feature works at the command-line script level and the same functionality is also supported in the menu-driven text console. It is not currently available through XenCenter.

When a metadata backup is first taken, a special backup VDI is created on a SR. This VDI has an ext3 filesystem which stores the following versioned backups:

- A full pool-database backup.
- Individual VM metadata backups, partitioned by the SRs in which the VM has disks.
- SR-level metadata which can be used to recreate the SR description when the storage is reattached.

In the menu-driven text console on the XenServer host, there are some menu items under the **Backup, Update and Restore** menu which provide more user-friendly interfaces to these scripts. The operations should only be performed on the pool master. You can use these menu items to perform 3 operations:

- Schedule a regular metadata backup to the default pool SR, either daily, weekly or monthly. This will regularly rotate metadata backups and ensure that the latest metadata is present for that SR without any user intervention being required.
- Trigger an immediate metadata backup to the SR of your choice. This will create a backup VDI if necessary, and attach it to the host and backup all the metadata to that SR. Use this option if you have made some changes which you want to see reflected in the backup immediately.
- Perform a metadata restoration operation. This will prompt you to choose an SR to restore from, and then the option of restoring only VM records associated with that SR, or all the VM records found (potentially from other SRs which were present at the time of the backup). There is also a 'dry run' option to see which VMs would be imported, but not actually perform the operation.
For automating this scripting, there are some commands in the control domain which provide an interface to metadata backup and restore at a lower level than the menu options:

- **xe-backup-metadata** which provides an interface to create the backup VDIs (with the -c flag), and also to attach the metadata backup and examine its contents.
- **xe-restore-metadata** which can be used to probe for a backup VDI on a newly attached SR, and also selectively reimport VM metadata to recreate the associations between VMs and their disks.

Full usage information for both scripts can be obtained by running them in the control domain with the \(-h\) flag. One particularly useful invocation mode is **xe-backup-metadata -d** which mounts the backup VDI into dom0, and drops into a sub-shell with the backup directory so it can be examined.

### Moving SRs between hosts and Pools

The metadata backup and restore options can be run as scripts within the control domain or through the **Backup, Restore, and Update** menu option in the Local Console. All other actions, such as detaching the SR from the source host and reattaching it to the destination host, can be performed using the Local Console, XenCenter, or the xe CLI. This example uses a combination of XenCenter and the Local Console.

#### To create and move a portable SR using Local Console and XenCenter

1. On the source host or pool, within the Local Console select the **Backup, Restore, and Update** menu option, select the **Backup Virtual Machine Metadata** option, and then select the desired SR:

2. Within XenCenter, select the source host or pool and shutdown all running VMs with VDIs on the SR to be moved.

3. Within the tree view select the SR to be moved and select **Storage > Detach Storage Repository**. The **Detach Storage Repository** menu option will not be displayed if there are running VMs with VDIs on the selected SR. After being detached the SR will be displayed in a grayed-out state.
Warning

Do not complete this step unless you have created a backup VDI in step 1.

4. Select **Storage > Forget Storage Repository** to remove the SR record from the host or pool.

5. Select the destination host in the tree view and select **Storage > New Storage Repository**.

6. Create a new SR with the appropriate parameters required to reconnect the existing SR to the destination host. In the case of moving a SR between pools or hosts within a site the parameters may be identical to the source pool.

7. Every time a new SR is created the storage is checked to see if it contains an existing SR. If so, an option is presented allowing re-attachment of the existing SR. If this option is not displayed the parameters specified during SR creation are not correct:

8. Select **Reattach**.

9. Select the new SR in the tree view and then select the **Storage** tab to view the existing VDIs present on the SR.

10. Within the Local Console on the destination host, select the **Backup, Restore, and Update** menu option, select the **Restore Virtual Machine Metadata** option, and select the newly re-attached SR.
11. The VDIs on the selected SR will be inspected to find the metadata VDI. Once found, select the desired metadata backup to use.

12. Select the **Only VMs on this SR** option to restore the VMs.

**Note**

Note: Use the **All VM Metadata** option when moving multiple SRs between hosts or pools, or when using tiered storage where VMs to be restored have VDIs on multiple SRs. When using this option ensure all required SRs have been reattached to the destination host prior running the restore.

13. The VMs will be restored in the destination pool in a shutdown state and are available for use.

**Using Portable SRs for Manual Multi-Site Disaster Recovery**

The Portable SR feature can be used in combination with storage layer replication in order to simplify the process of creating and enabling a disaster recovery (DR) site. Using storage layer replication to mirror or replicate LUNs that comprise portable SRs between production and DR sites allows all required data to be automatically present in the DR site. The constraints that apply when moving portable SRs between hosts or pools within the same site also apply in the multi-site case, but the production and DR sites are not required to have the same number of hosts. This allows use of either dedicated DR facilities or non-dedicated DR sites that run other production workloads.

**Using portable SRs with storage layer replication between sites to enable the DR site in case of disaster**

1. Any storage layer configuration required to enable the mirror or replica LUN in the DR site are performed.
2. A SR is created for each LUN in the DR site.
3. VMs are restored from metadata on one or more SRs.
4. Any adjustments to VM configuration required by differences in the DR site, such as IP addressing, are performed.
5. VMs are started and verified.
6. Traffic is routed to the VMs in the DR site.

**VM Snapshots**

XenServer 5.0.0 provides a convenient snapshotting mechanism that can take a snapshot of a VM's storage and metadata at a given time. Where necessary IO is temporarily halted while the snapshot is being taken to ensure that a self-consistent disk image can be captured.

Snapshot operations result in a snapshot VM that is similar to a template. The VM snapshot contains all the storage information and VM configuration, including attached VIFs, allowing them to be exported and restored for backup purposes.

The snapshotting operation is a 2 step process:

- Capturing metadata as a template.
- Issuing a VDI snapshot of the disk(s).

Two types of VM snapshots are supported: regular and quiesced:

**Regular Snapshots**

Regular snapshots are crash consistent and can be performed on all VM types, including Linux VMs.

**Quiesced Snapshots**

Quiesced snapshots are a special case that take advantage of the Windows Volume Snapshot Service (VSS) for services that support it, so that a supported application (for example Microsoft Exchange or SQLServer) can flush data to disk and prepare for the snapshot before it is taken.

Quiesced snapshots are therefore safer to restore, but can have a greater performance impact on a system while they are being taken. They may also fail under load so more than one attempt to take the snapshot may be required.

**Warning**

VDIs that have been snapshotted from outside the VM can be accessed by the VM via the VSS request interface.

XenServer supports quiesced snapshots on Windows Server 2003 and Windows Server 2008 for both 32-bit and 64-bit variants which have Microsoft VSS installed. Windows 2000, Windows XP and Windows Vista are not supported. Supported storage backends are EqualLogic and NetApp.

**Note**

Don't forget to install the Xen VSS provider in the Windows guest in order to support VSS. This is done via the install-XenProvider.cmd script provided with the Windows PV drivers. More details can be found in the Virtual Machine Installation Guide under the Windows section.

In general, a VM can only access VDI snapshots (not VDI clones) of itself via the VSS interface. There is a flag that can be set by the XenServer administrator whereby adding an attribute of snapmanager=true to the VM's
other-config allows that VM to import snapshots of VDIs from other VMs. Note that this opens a security vulnerability and should be used with care. This feature allows an administrator to attach VSS snapshots using an in-guest transportable snapshot ID as generated by the VSS layer to another VM for the purposes of backup.

VSS quiesce timeout: the Microsoft OS quiesce period is limited to only 10 seconds, therefore it is quite feasible that a snapshot may not be able to complete in time. If, for example the XAPI daemon has queued additional blocking tasks such as an SR scan, the VSS snapshot may timeout and fail. The operation should be retried if this happens. Note also that the larger the number of VBDs attached to a VM, the more likely it is that this timeout may be reached.

VSS snap of entire VM's disks: in order to store all data available at the time of a VSS snapshot, the XAPI manager will snapshot all disks plus VM metadata associated with a VM that are snapshottable via the Xen storage manager API. If the VSS layer requests a subset of the disks, you will notice that a full VM snapshot is taken regardless. The extra snapshot data can be deleted via XenCenter if not required.

**Taking a VM snapshot**

Before taking a snapshot, see the section called “Preparing to clone a Windows VM” in XenServer Virtual Machine Installation Guide and the section called “Preparing to clone a Linux VM” in XenServer Virtual Machine Installation Guide for information about any special operating system-specific configuration and considerations to take into account.

Use the `vm-snapshot` and `vm-snapshot-with-quiesce` commands to take a snapshot of a VM:

```bash
xe vm-snapshot vm=<vm_name> new-name-label=<vm_snapshot_name>
xe vm-snapshot-with-quiesce vm=<vm_name> new-name-label=<vm_snapshot_name>
```

**VM Rollback**

**Restoring a VM to snapshot state**

1. Note the name of the snapshot
2. Note the MAC address of the VM
3. Destroy the VM:
   a. Run the `vm-list` command to find the UUID of the VM to be destroyed:
      ```bash
      xe vm-list
      ```
   b. Shutdown the VM:
      ```bash
      xe vm-shutdown uuid=<vm_uuid>
      ```
   c. Destroy the VM:
      ```bash
      xe vm-destroy uuid=<vm_uuid>
      ```
4. Note the name of the storage to restore the VM on:
   ```bash
   xe sr-list
   ```
   The `sr-list` command lists all storage in the pool or on the host.
5. Create a new VM from the snapshot:
   ```bash
   xe vm-install new-name-label=<vm_name_label> template=<template_name> \
   sr-name-label=<sr_name>
   ```
6. Edit the MAC address of the VM to be the same as the MAC address of the VM begin restored.
7. Start the VM:

```bash
xe vm-start name-label=<vm_name>
```

---

**Coping with machine failures**

This section provides details of how to recover from various failure scenarios. All failure recovery scenarios require the use of one or more of the backup types listed in the section called “Backups”.

**Member failures**

In the absence of HA, master nodes detect the failures of members by receiving regular heartbeat messages. If no heartbeat has been received for 200 seconds, the master assumes the member is dead. There are two ways to recover from this problem:

- Repair the dead host (e.g. by physically rebooting it). When the connection to the member is restored, the master will mark the member as alive again.
- Shutdown the host and instruct the master to forget about the member node using the `xe host-forget` CLI command. Once the member has been forgotten, all the VMs which were running there will be marked as offline and can be restarted on other XenServer hosts. Note it is **very** important to ensure that the XenServer host is actually offline, otherwise VM data corruption might occur. Be careful not to split your pool into multiple pools of a single host by using `xe host-forget`, since this could result in them all mapping the same shared storage and corrupting VM data.

**Warning**

You should reinstall the XenServer software on a host that has been forgotten.

When a member XenServer host fails, there may be VMs still registered in the **running** state. If you are sure that the member XenServer host is definitely down, and that the VMs have not been brought up on another XenServer host in the pool, use the `xe vm-reset-powerstate` CLI command to set the power state of the VMs to **halted**. See the section called “vm-reset-powerstate” for more details.

**Warning**

Incorrect use of this command can lead to data corruption. Only use this command if absolutely necessary.

**Master failures**

Every member of a resource pool contains all the information necessary to take over the role of master if required. When a master node fails, the following sequence of events occurs:

1. The members realize that communication has been lost and each retry for sixty seconds
2. Each member then puts itself into emergency mode, whereby the member XenServer hosts will now accept only the pool-emergency commands (`xe pool-emergency-reset-master` and `xe pool-emergency-transition-to-master`).

If the master comes back up at this point, it will re-establish communication with its members. The members will leave emergency mode and operation will return to normal.
If the master is really dead, though, you should choose one of the members and issue to it the command `xe pool-emergency-transition-to-master`. Once it has become the master, issue the command `xe pool-recover-slaves` and the members will now point to the new master.

If you repair or replace the server that was the original master, you can simply bring it up, install the XenServer host software, and add it to the pool. Since the XenServer hosts in the pool are enforced to be homogeneous, there is no real need to make the replaced server the master.

When a member XenServer host is transitioned to being a master, you should also check that the default pool storage repository is set to an appropriate value. This can be done using the `xe pool-param-list` command and verifying that the `default-SR` parameter is pointing to a valid storage repository.

**Pool failures**

In the unfortunate event that your entire resource pool fails, you will need to recreate the pool database from scratch. Be sure to regularly back up your pool-metadata using the `xe pool-dump-database` CLI command (see the section called “pool-dump-database”).

**To restore a completely failed pool**

1. Install a fresh set of hosts, and apply your Enterprise Edition license to them. Do not pool them up at this stage.
2. For the host nominated as the master, restore the pool database from your backup via the `xe pool-restore-database` (see the section called “pool-restore-database”) CLI command.
3. Connect to the master host via XenCenter and ensure that all your shared storage and VMs are available again.
4. Perform a pool join operation on the remaining freshly installed member hosts, and start up your VMs on the appropriate hosts.

**Coping with Failure due to Configuration Errors**

If the physical host machine is operational but the software or host configuration is corrupted:

**To restore host software and configuration**

1. Run the command:

   ```
   xe host-restore host=<host> file-name=<hostbackup>
   ```

2. Reboot to the host installation CD and select **Restore from backup**.

**Physical Machine failure**

If the physical host machine has failed, use the appropriate procedure listed below to recover.

---

**Warning**

Any VMs which were running on a previous member (or the previous host) which has failed will still be marked as Running in the database. This is for safety -- simultaneously starting a VM on two different hosts would lead to severe disk corruption. If you are sure that the machines (and VMs) are offline you can reset the VM power state to Halted:

```
xe vm-reset-powerstate vm=<vm><uid> --force
```
VMs can then be restarted using XenCenter or the CLI.

### Replacing a failed master with a still running member

1. Run the commands:

   ```
   xe pool-emergency-transition-to-master
   xe pool-recover-slaves
   ```

2. If the commands succeed, restart the VMs.

### To restore a pool with all hosts failed

1. Run the command:

   ```
   xe pool-restore-database file-name=<backup>
   ```

**Warning**

This command will only succeed if the target machine has an appropriate number of appropriately named NICs.

2. If the target machine has a different view of the storage (for example, a block-mirror with a different IP address) than the original machine, modify the storage configuration using `pbd-destroy` and `pbd-create` to recreate storage configurations. See the section called “PBD commands” for documentation of these commands.

3. If you have created a new storage configuration, use `pbd-plug` or Storage > Repair Storage Repository menu item to use the new configuration.

4. Restart all VMs.

### To restore a VM when VM storage is not available

1. Run the command:

   ```
   xe vm-import filename=<backup> --metadata
   ```

2. If the metadata import fails, run the command:

   ```
   xe vm-import filename=<backup> --metadata --force
   ```

   This command will attempt to restore the VM metadata on a 'best effort' basis.

3. Restart all VMs.
Chapter 3. Storage

This chapter discusses the framework for storage abstractions. It describes the way physical storage hardware of various kinds is mapped to VMs, and the software objects used by the XenServer host API to perform storage-related tasks. Detailed sections on each of the supported storage types include procedures for creating storage for VMs using the CLI, with type-specific device configuration options, and some best practices for managing storage in XenServer host environments. Finally, the virtual disk QoS (quality of service) settings available to XenServer Enterprise Edition are described.

Overview

Storage repositories (SRs)

XenServer host defines a container called a storage repository (SR) to describe a particular storage target, in which Virtual Disk Images (VDIs) are stored. A VDI is a disk abstraction which contains the contents of a virtual disk.

The interface to storage hardware allows VDIs to be supported on a large number of SR types. With built-in support for IDE, SATA, SCSI and SAS drives locally connected, and iSCSI, NFS and Fibre Channel remotely connected, the XenServer host SR is very flexible. The SR and VDI abstractions allows advanced storage features such as sparse provisioning, VDI snapshots, and fast cloning to be exposed on storage targets that support them. For storage subsystems that do not inherently support advanced operations directly, a software stack is provided based on Microsoft's Virtual Hard Disk (VHD) specification which implements these features.

Each XenServer host can use multiple SRs and different SR types simultaneously. These SRs can be shared, or dedicated between hosts. As mentioned previously in Chapter 2, XenServer hosts and resource pools, shared storage is pooled between multiple hosts within a defined resource pool. A Shared SR must be network accessible to each host, and thus must be an iSCSI, NFS, NetApp, EqualLogic or Fibre Channel SR. Finally, all hosts in a single resource pool must have at least one shared SR in common.

Virtual Disk Images (VDI)

There are four basic VDI types, VHD, Logical Volume Manager (LVM), EqualLogic, and NetApp managed LUNs. Both VHD and LVM types can exist on local dedicated storage or remote shared storage.

- The VHD format can be used on either a local disk with an ext3 filesystem or on an NFS share. VDIs stored in VHD format are sparse. The image file is automatically extended in 2MB chunks as the VM writes data into the disk. This has the considerable benefit that VM image files take up only as much space on the physical storage as required. If a 100GB VDI is allocated for a new VM and an OS is installed, the VDI file will physically be only the size of the OS data that has been written to the disk, plus some minor metadata overhead.

VDH files may also be chained, allowing two VDIs to share common data. In cases where a VHD-backed VM is cloned, the resulting VMs share the common on-disk data at the time of cloning. Each proceeds to make its own changes in an isolated copy-on-write version of the VDI. This feature allows VHD-based VMs to be quickly cloned from templates, facilitating very fast provisioning and deployment of new VMs.

When cloning VMs based off a single VHD template, each child VM forms a "chain" where new changes are written to the new VM, and old blocks are directly read from the parent template. If the new VM were to be converted into a further template and more VMs cloned, then the resulting chain will result in degraded performance. XenServer supports a maximum chain length of 30, but it is generally not recommended that you approach this limit without good reason. If in doubt, you can always "copy" the VM using the xe vm-copy command or using XenServer, which will reset the chain length back to 0.
• The LVM format can be used on either local disk or shared storage. Shared LVM access can be provided using either an iSCSI or a Fibre Channel LUN hosted on a network array that allows shared access by multiple initiators or HBAs.

• Managed NetApp LUNs are accessible via the NetApp SR driver type, and are hosted on a Network Appliance device running a version of Ontap 7.0 or greater. LUNs are allocated and mapped dynamically to the host via the XenServer host management framework.

• EqualLogic storage is accessible via the EqualLogic SR driver type, and are hosted on an EqualLogic storage array. LUNs are allocated and mapped dynamically to the host via the XenServer host management framework.

**Managing Storage**

In working with the XenServer host CLI, there are four object classes that are used to describe, configure, and manage storage:

• Storage Repositories (SRs) are storage targets containing homogeneous virtual disks (VDIs). SR commands provide operations for creating, destroying, resizing, cloning, connecting and discovering the individual Virtual Disk Images (VDIs) that they contain.

A storage repository is a persistent, on-disk data structure. So the act of "creating" a new SR is similar to that of formatting a disk -- for most SR types, creating a new SR involves erasing any existing data on the specified storage target. The exception is NFS SRs, which create a new directory on the filer leaving existing NFS SRs as they were. SRs are long-lived, and may in some cases be shared among XenServer hosts, or moved between them. CLI operations to manage storage repositories are described in the section called "SR commands".

• Physical Block Devices (PBDs) represent the interface between a physical server and an attached SR. PBDs are connector objects that allow a given SR to be mapped to a XenServer host. PBDs store the device configuration fields that are used to connect to and interact with a given storage target. For example, NFS device configuration includes the IP address of the NFS server and the associated path that the XenServer host mounts. PBD objects manage the run-time attachment of a given SR to a given XenServer host. CLI operations relating to PBDs are described in the section called “PBD commands”.

• Virtual Disk Images (VDIs) are an on-disk representation of a virtual disk provided to a VM. VDIs are the fundamental unit of virtualized storage in XenServer.

Similar to SRs, VDIs are persistent, on-disk objects that exist independently of XenServer hosts. CLI operations to manage VDIs are presented in the section called “VDI commands”.

• Virtual Block Devices (VBDs) are a connector object (similar to the PBD described above) that allows mappings between VDIs and Virtual Machines (VMs). In addition to providing a mechanism to attach (or plug) a VDI into a VM, VBDs allow the fine-tuning of parameters regarding QoS (quality of service), statistics, and the bootability of a given VDI. CLI operations relating to VBDs are described in the section called “VBD commands”.

Graphical overview of storage repositories and related objects

Storage Repository Types

This section provides descriptions of the physical storage types that XenServer supports. Device configuration options and examples of creating SRs are given for each type.

<table>
<thead>
<tr>
<th>SR type</th>
<th>Description</th>
<th>Shared?</th>
<th>Sparse?</th>
<th>VDI Re-size?</th>
<th>Fast Clone?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext</td>
<td>VHD on Local Disk</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>nfs</td>
<td>VHD on Network File System</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>lvm</td>
<td>Logical Volume Management on Local Disk</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>lvmohba</td>
<td>Logical Volume Management over Fibre Channel or iSCSI Host Bus Adapter (HBA)</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>lvmiscsi</td>
<td>Logical Volume Management over iSCSI using software initiator</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>netapp</td>
<td>NetApp filer using Ontap</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>equal</td>
<td>EqualLogic filer</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

The storage repository types supported in XenServer are provided by plug-ins in the control domain; these can be examined and plugins supported by third parties may be added to the /opt/xensource/sm directory. Modification of these files is unsupported, but visibility of these files may be valuable to developers and power users. New storage manager plugins can be placed in this directory and will be automatically detected by XenServer. The available SR types may be listed using the sm-list command (see the section called “Storage Manager commands”).
New storage repositories are created using via the GUI using the New Storage wizard that will guide you through the various probing and configuration steps, or by using the `sr-create` CLI command. This command creates a new SR on the storage substrate (potentially destroying any existing data), and makes the storage repository API object and a corresponding PBD records, allowing use of the storage by virtual machines. Upon successful creation of the SR, the PBD will be automatically plugged. If the SR shared=true flag is set, a PBD entry will be created and plugged for every XenServer Host in the resource pool.

**Local Disks**

By default, XenServer uses the local disk on the physical host on which it is installed. The Linux Logical Volume Manager (LVM) is used to manage VM storage. In this case a VDI is implemented as an LVM logical volume of the specified size.

Local LVM-based storage is high-performance and allows virtual disks to be dynamically resized. Virtual disks are fully allocated as an isolated volume on the underlying physical disk and so there is a minimum of storage virtualization overhead imposed. As such, this is a good option for high-performance storage, but lacks the flexibility of file-based storage options described below.

In addition to storing virtual disks on an LVM-managed volume, local disks may be configured with a local EXT SR to serve VDIs stored in the Microsoft VHD format. Local disk EXT SRs must be configured via the XenServer CLI.

By definition, local disks are not shared across pools of XenServer host. As a consequence, VMs whose VDIs are stored in SRs on local disks are not agile -- they may not be migrated between XenServer hosts in a resource pool.

**Local hotplug devices**

XenServer has two SRs of type `udev` that represent removable storage. One is for the CD or DVD disk in the physical CD or DVD-ROM drive of the XenServer host. The other is for a USB read/write device plugged into a USB port of the XenServer host. VDIs that represent the media come and go as disks or USB sticks are inserted and removed.

**Shared Network Attached Storage using NFS**

The NFS filer is a ubiquitous form of storage infrastructure that is available in many environments. XenServer allows existing NFS servers that support NFS V3 over TCP/IP to be used immediately as a storage repository for virtual disks (VDIs). VDIs are stored in the Microsoft VHD format only. Moreover, as NFS SRs can be shared, VDIs stored in a shared SR allow VMs to be started on any XenServer hosts in a resource pool and be migrated between them using XenMotion with no noticeable downtime.

Creating an NFS SR requires the hostname or IP address of the NFS server. The `sr-probe` command can provide a list of valid destination paths exported by the server on which the SR may be created. The NFS server must be configured to export the specified path to all XenServer hosts in the pool, or the creation of the SR and the plugging of the PBD record will fail.

As mentioned at the beginning of this chapter, VDIs stored on NFS are sparse. The image file is allocated as the VM writes data into the disk. This has the considerable benefit that VM image files take up only as much space on the NFS filer as is required. If a 100GB VDI is allocated for a new VM and an OS is installed, the VDI file will only reflect the size of the OS data that has been written to the disk rather than the entire 100GB.

VHD files may also be chained, allowing two VDIs to share common data. In cases where a NFS-based VM is cloned, the resulting VMs will share the common on-disk data at the time of cloning. Each will proceed to make its own changes in an isolated copy-on-write version of the VDI. This feature allows NFS-based VMs to be quickly cloned from templates, facilitating very fast provisioning and deployment of new VMs.
As VHD-based images require extra metadata to support sparseness and chaining, the format is not as high-performance as LVM-based storage. In cases where performance really matters, it is well worth forcibly allocating the sparse regions of an image file. This will improve performance at the cost of consuming additional disk space.

XenServer's NFS and VHD implementation assume that they have full control over the SR directory on the NFS server. Administrators should not modify the contents of the SR directory, as this can risk corrupting the contents of VDIs.

For NFS best practices, consider performance and access control issues. Access control can specify which client IP address has access to an NFS export. Alternatively, one can allow world read/write access control. The administrator ought to make policy decisions based on the specific requirements of their installation.

XenServer has been tuned for enterprise-class filers that use non-volatile RAM to provide fast acknowledgments of write requests while maintaining a high degree of data protection from failure. For reference, XenServer has been tested extensively against Network Appliance FAS270c and FAS3020c filers, using Data OnTap 7.2.2.

In situations where XenServer is used with lower-end filers, it will err on the side of caution by waiting for all writes to be acknowledged before passing acknowledgments on to guest VMs. This will incur a noticeable performance cost, and might be remedied by setting the filer to present the SR mount point as an asynchronous mode export. Asynchronous exports acknowledge writes that are not actually on disk, and so administrators should consider the risks of failure carefully in these situations.

The XenServer NFS implementation uses TCP by default. If your situation allows, you can configure the implementation to use UDP in situations where there may be a performance benefit. To do this, specify the `device-config` parameter `useUDP=true` at SR creation time.

---

**Warning**

Since VDIs on NFS SRs are created as sparse, administrators must ensure that there is enough disk space on the NFS SRs for all required VDIs. XenServer hosts do not enforce that the space required for VDIs on NFS SRs is actually present.

---

**Shared iSCSI Storage**

XenServer provides support for shared SRs on iSCSI LUNs. iSCSI is supported using the open-iSCSI software iSCSI initiator or by using a supported iSCSI Host Bus Adapter (HBA). The steps for using iSCSI HBAs are identical to those for Fibre Channel HBAs, both of which are described in the section called “Shared LVM storage over FC or iSCSI hardware HBAs”.

Shared iSCSI support using the software iSCSI initiator is implemented based on the Linux Volume Manager (LVM) and provides the same performance benefits provided by LVM VDIs in the local disk case. Shared iSCSI SRs using the software-based host initiator are capable of supporting VM agility using XenMotion: VMs can be started on any XenServer host in a resource pool and migrated between them with no noticeable downtime. The LVM VDIs used in software-based iSCSI SRs VDIs do not provide support for sparse provisioning or fast cloning.

iSCSI SRs utilize the entire LUN specified at creation time and may not span more than one LUN. CHAP support is provided for client authentication, during both the data path initialization and the LUN discovery phases.

**XenServer Host iSCSI configuration**

All iSCSI initiators and targets must have a unique name to ensure they can be uniquely identified on the network. An initiator has an iSCSI initiator address, and a target has an iSCSI target address. Collectively these are called iSCSI Qualified Names, or IQNs.
XenServer hosts support a single iSCSI initiator which is automatically created and configured with a random IQN during host installation. The single initiator can be used to connect to multiple iSCSI targets concurrently.

iSCSI targets commonly provide access control via iSCSI initiator IQN lists, so all iSCSI targets/LUNs to be accessed by a XenServer host must be configured to allow access by the host's initiator IQN. Similarly, targets/LUNs to be used as shared iSCSI SRs must be configured to allow access by all host IQNs in the resource pool.

**Note**

iSCSI targets that do not provide access control will typically default to restricting LUN access to a single initiator to ensure data integrity. If an iSCSI LUN is intended for use as a shared SR across multiple XenServer hosts in a resource pool ensure that multi-initiator access is enabled for the specified LUN.

The XenServer host IQN value can be adjusted using XenCenter, or via the CLI with the following command when using the iSCSI software initiator:

```
xe host-param-set uuid=<valid_host_id> other-config:iscsi_iqn=<new_initiator_iqn>
```

**Warning**

It is imperative that every iSCSI target and initiator have a unique IQN. If a non-unique IQN identifier is used, data corruption and/or denial of LUN access can occur.

**Warning**

Do not change the XenServer host IQN with iSCSI SRs attached. Doing so can result in failures connecting to new targets or existing SRs.

**Shared LVM storage over FC or iSCSI hardware HBAs**

XenServer hosts support Fibre Channel (FC) storage area networks (SANs) through Emulex or QLogic host bus adapters (HBAs). All FC configuration required to expose a FC LUN to the host must be completed manually, including storage devices, network devices, and the HBA within the XenServer host. Once all FC configuration is complete the HBA will expose a SCSI device backed by the FC LUN to the host. The SCSI device can then be used to access to the FC LUN as if it were a locally attached SCSI device.

The `sr-probe` command should be used to list the LUN-backed SCSI devices present on the host, and will force a scan for new LUN-backed SCSI devices each time it is run. The path value returned by `sr-probe` for a LUN-backed SCSI device will be consistent across all hosts with access to the LUN, and therefore must be used when creating shared SRs accessible by all hosts in a resource pool.

The same features apply to QLogic iSCSI HBAs.

See the section called “Creating Storage Repositories” for details on creating shared HBA-based FC and iSCSI SRs.
**Note**

XenServer support for Fibre Channel does not support direct mapping of a LUN to a VM. HBA-based LUNs must be mapped to the host and specified for use in an SR. VDIs within the SR are exposed to VMs as standard block devices.

**Shared NetApp Storage**

If you have access to a Network Appliance™ (NetApp) filer with sufficient disk space, running a version of Data ONTAP 7G (version 7.0 or greater), you can configure a custom NetApp storage repository for VM storage on your XenServer deployment. The XenServer driver uses the ZAPI interface to the filer to create a group of FlexVols which correspond to an SR. VDIs are created as virtual LUNs on the filer, and attached to XenServer hosts using an iSCSI data path. There is a direct mapping between a VDI and a raw LUN without requiring any additional volume metadata. Thus, at a logical level, the NetApp SR is a managed volume and the VDIs are the LUNs within the volume. VM cloning uses the snapshotting and cloning capabilities of the filer for data efficiency and performance and to ensure compatibility with existing ONTAP filer management tools.

As with the iSCSI-based SR type, the NetApp driver also uses the built-in software initiator and its assigned server IQN, which can be modified by changing the value shown on the General tab when the storage repository is selected in XenCenter.

The simplest way to create NetApp SRs is with XenCenter. See XenCenter Help for details. They can also be created using xe CLI commands. See the section called “Creating a shared NetApp SR over iSCSI” for an example.

**FlexVols**

NetApp introduces a notion of FlexVol as the basic unit of manageable data. It is important to note that there are limitations that constrain the design of NetApp-based SRs. These are:

- maximum number of FlexVols per filer
- maximum number of LUNs per network port
- maximum number of snapshots per FlexVol

Precise system limits vary per Filer type, however as a general guide, a FlexVol may contain up to 200 LUNs, and provides up to 255 snapshots. Since there is a one-to-one mapping of LUNs to VDIs, and often a VM will have more than one VDI, it is apparent that the resource limitations of a single FlexVol can easily be reached. Also, consider that the act of taking a snapshot includes snapshotting all the LUNs within a FlexVol and that the VM clone operation indirectly relies on snapshots in the background as well as the CLI-based VDI snapshot operation for backup purposes.

There are two constraints to consider, therefore, in mapping the virtual storage objects of the XenServer host to the filer; in order to maintain space efficiency it makes sense to limit the number of LUNs per FlexVol, yet at the other extreme, in order to avoid resource limitations a single LUN per FlexVol provides the most flexibility. However, since there is a vendor-imposed limit of 200 or 500 FlexVols, per filer (depending on the NetApp model), this would create a limit of 200 or 500 VDIs per filer and it is therefore important to select a suitable number of FlexVols around these parameters.

Given these resource constraints, the mapping of virtual storage objects to the Ontap storage system has been designed in the following manner; LUNs are distributed evenly across FlexVols, with the expectation of using VM UUIDs to opportunistically group LUNs attached to the same VM into the same FlexVol. This is a reasonable usage model that allows a snapshot of all the VDIs in a VM at one time, maximizing the efficiency of the snapshot operation.

An optional parameter you can set is the number of FlexVols assigned to the SR. You can use between 1 and 32 FlexVols; the default is 8. The trade-off in the number of FlexVols to the SR is that, for a greater number of FlexVols,
the snapshot and clone operations become more efficient, since there are statistically fewer VMs backed off the same FlexVol. The disadvantage is that more FlexVol resources are used for a single SR, where there is a typical system-wide limitation of 200 for some smaller filers.

**Aggregates**

When creating a NetApp driver-based SR, you select an appropriate aggregate. The driver can be probed for non-traditional type aggregates, that is, newer-style aggregates that support FlexVols, and then lists all aggregates available and the unused disk space on each.

**Note**

Note that aggregate probing is only possible at sr-create time. It needs to be done there so that the aggregate can be specified at the point that the SR is created, but is not probed by the sr-probe command.

We strongly recommend that you configure an aggregate exclusively for use by XenServer storage, since space guarantees and allocation cannot be correctly managed if other applications are also sharing the resource.

**Thick or thin provisioning**

When creating NetApp storage, you can also choose the type of space management used. By default, allocated space is “thickly provisioned” to ensure that VMs never run out of disk space and that all virtual allocation guarantees are fully enforced on the filer. Selecting “thick provisioning” ensures that whenever a VDI (LUN) is allocated on the filer, sufficient space is reserved to guarantee that it will never run out of space and consequently experience failed writes to disk. Due to the nature of the Ontap FlexVol space provisioning algorithms the best practice guidelines for the filer require that at least twice the LUN space is reserved to account for background snapshot data collection and to ensure that writes to disk are never blocked. In addition to the double disk space guarantee, Ontap also requires some additional space reservation for management of unique blocks across snapshots. The guideline on this amount is 20% above the reserved space. Therefore, the space guarantees afforded by “thick provisioning” will reserve up to 2.4 times the requested virtual disk space.

The alternative allocation strategy is thin provisioning, which allows the admin to present more storage space to the VMs connecting to the SR than is actually available on the SR. There are no space guarantees, and allocation of a LUN does not claim any data blocks in the FlexVol until the VM writes data. This would be appropriate for development and test environments where you might find it convenient to over-provision virtual disk space on the SR in the anticipation that VMs may be created and destroyed frequently without ever utilizing the full virtual allocated disk. This method should be used with extreme caution and only in non-critical environments.

**FAS Deduplication**

FAS Deduplication is a NetApp technology for reclaiming redundant disk space. Newly-stored data objects are divided into small blocks, each block containing a digital signature, which is compared to all other signatures in the data volume. If an exact block match exists, the duplicate block is discarded and the disk space reclaimed. FAS Deduplication can be enabled on “thin provisioned” NetApp-based SRs and will operate according to the default filer FAS Deduplication parameters, typically every 24 hours. It must be enabled at the point the SR is created and any custom FAS Deduplication configuration managed directly on the filer.

**Access Control**

Since FlexVol operations such as volume creation and volume snapshotting require administrator privileges on the filer itself, it is recommended that the XenServer host should be provided with suitable administrator username and
password credentials at configuration time. In situations where the XenServer host does not have full administrator rights to the filer, the filer administrator could perform an out-of-band preparation and provisioning of the filer. The SR is then introduced to the XenServer host using the XenCenter or via the sr-introduce xe CLI command. Note however that operations such as VM cloning or snapshot generation will fail in this situation due to insufficient access privileges.

**Licenses**

You need to have an iSCSI license on the NetApp filer to use this storage repository type; for the generic plugins you will required either an iSCSI or NFS license depending on the SR type being used.

**Further information**

For more information about NetApp technology, see the following links:

- General information on NetApp products
- Data ONTAP
- FlexVol
- FlexClone
- RAID-DP
- Snapshot
- FilerView

**Shared EqualLogic Storage**

If you have access to an EqualLogic filer, you can configure a custom EqualLogic storage repository for VM storage on your XenServer deployment. This will allow use of the advanced features of this filer type, and will store virtual disks on the filer using one LUN per virtual disk. Using this storage type will enable the thin provisioning, snapshot, and fast clone features of this filer.

You should consider your storage requirements when deciding whether to use the specialized SR plugin, or to use the generic LVM/iSCSI storage backend. By using the specialized plugin, XenServer will communicate with the filer to provision storage. Note that some arrays have a limitation of seven concurrent connections, which may limit the throughput of control operations. Using the plugin will allow you to make use of the advanced array features, however, so will make backup and snapshot operations easier.

**Warning**

There are two types of administration accounts that can successfully access the EqualLogic SM plugin:

- A group administration account which has access to and can manage the entire group and all storage pools.
- A pool administrator account that can manage only the objects (SR’s and VDI’s/Snapshots) that are in the pool or pools assigned to the account.

**Storage configuration examples**

This section covers creating storage repository types and making them available to a XenServer host. The examples provided pertain to storage configuration via the CLI which provides the greatest flexibility. See the XenCenter Help for details on using the **New Storage Repository** wizard.
Creating Storage Repositories

This section covers creating Storage Repositories (SRs) of different types and making them available to a XenServer host. The examples provided cover creating Storage Repositories (SRs) via the xe CLI. See the XenCenter Help for details on using the New Storage Repository wizard to add SRs via XenCenter.

**Note**

Local SRs of type lvm and ext and HBA-based FC and iSCSI SRs of type lvmohba can only be created with the CLI. After creation all SR types can be managed by either XenCenter or the CLI.

There are two basic steps involved in creating a new storage repository for use on a XenServer host via the CLI:

1. Probe the SR type to determine values for any required parameters.
2. Create the SR to create the SR object and associated PBD objects, plug the PBDs, and activate the SR.

These steps differ in detail depending on type of SR being created. In all examples the sr-create command will return the UUID of the SR if successful.

SRs can also be destroyed when no longer in use to free up the physical device, or forgotten to detach the SR from one XenServer host and attach it to another. See the section called “Destroying or forgetting a SR” for details.

Probing an SR

The `sr-probe` CLI command can be used in two ways:

1. To identify unknown parameters for use in creating a SR
2. To return a list of existing SRs

In both cases `sr-probe` works by specifying an SR type and one or more `device-config` parameters for that SR type. When an incomplete set of parameters is supplied `sr-probe` returns an error message indicating parameters are missing and the possible options for the missing parameters. When a complete set of parameters is supplied a list of existing SRs is returned. All `sr-probe` output is returned as an XML list.

For example, a known iSCSI target can be probed by specifying its name or IP address, and the set of IQNs available on the target will be returned:
xe sr-probe type=lvmiscsi device-config:target=192.168.1.10

Error code: SR_BACKEND_FAILURE_96
Error parameters: , The request is missing or has an incorrect target IQN parameter,

<?xml version="1.0" ?>
<iscsi-target-iqns>
  <TGT>
    <Index>0</Index>
    <IPAddress>192.168.1.10</IPAddress>
    <TargetIQN>iqn.192.168.1.10:filer1</TargetIQN>
  </TGT>
</iscsi-target-iqns>

Probing the same target again and specifying both the name/IP address and desired IQN will return the set of SCSIids (LUNs) available on the target/IQN.

xe sr-probe type=lvmiscsi device-config:target=192.168.1.10  \
  device-config:targetIQN=iqn.192.168.1.10:filer1

Error code: SR_BACKEND_FAILURE_107
Error parameters: , The SCSIid parameter is missing or incorrect,

<?xml version="1.0" ?>
<iscsi-target>
  <LUN>
    <vendor>IET</vendor>
    <LUNid>0</LUNid>
    <size>42949672960</size>
    <SCSIid>1494554000000000000000020000000b70200000f000000</SCSIid>
  </LUN>
</iscsi-target>

Probing the same target and supplying all three parameters will return a list of SRs that exist on the LUN, if any.
xe sr-probe type=lvmiscsi device-config:target=192.168.1.10  \
device-config:targetIQN=192.168.1.10:filer1  
device-config:SCSIid=14945540000000000000002000000b70200000f000000

<?xml version="1.0" ?>
<SRlist>
  <SR>
    <UUID>
      3f6e1ebd-8687-0315-f9d3-b02ab3adc4a6
    </UUID>
    <Devlist>
      /dev/disk/by-id/scsi-149455400000000000000000002000000b70200000f000000
    </Devlist>
  </SR>
</SRlist>

The following parameters can be probed for each SR type:

<table>
<thead>
<tr>
<th>SR type</th>
<th>device-config parameter, in order of dependency</th>
<th>Can be probed?</th>
<th>Required for sr-create?</th>
</tr>
</thead>
<tbody>
<tr>
<td>lvmoiscsi</td>
<td>target</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>chapuser</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>chappassword</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>targetIQN</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>SCSIid</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>lvmohba</td>
<td>SCSIid</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>netapp</td>
<td>target</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>username</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>password</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>chapuser</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>chappassword</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>aggregate</td>
<td>No*</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>FlexVols</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>allocation</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>asis</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>nfs</td>
<td>server</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>serverpath</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>lvm</td>
<td>device</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Creating a local LVM SR (lvm)

Device-config parameters for lvm SRs are:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>device name on the local host to use for the SR</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To create a local lvm SR on `/dev/sdb` use the following command.

```
xe sr-create host-uuid=<valid_uuid> content-type=user \
name-label="Example Local LVM SR" shared=false \
device-config:device=/dev/sdb type=lvm
```

Creating a local EXT3 SR (ext)

Device-config parameters for ext SRs:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>device name on the local host to use for the SR</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To create a local ext SR on `/dev/sdb` use the following command.

```
xe sr-create host-uuid=<valid_uuid> content-type=user \
name-label="Example Local EXT3 SR" shared=false \
device-config:device=/dev/sdb type=ext
```
Creating a shared NFS SR (nfs)

Device-config parameters for nfs SRs:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>server</td>
<td>IP address or hostname of the NFS server</td>
<td>Yes</td>
</tr>
<tr>
<td>serverpath</td>
<td>path, including the NFS mount point, on the NFS server in which the SR will reside</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To create a shared nfs SR on 192.168.1.10:/export1 use the following command.

```
xe sr-create host-uuid= <host_uuid> content-type=user 
name-label= "Example shared NFS SR" shared=true 
device-config:server=192.168.1.10 device-config:serverpath=/export1 type=nfs
```

Creating a shared LVM over iSCSI SR using the software iSCSI initiator (lvmoiscsi)

Device-config parameters for lvmoiscsi SRs:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Optional?</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>the IP address or hostname of the iSCSI filer in which the SR will reside</td>
<td>yes</td>
</tr>
<tr>
<td>targetIQN</td>
<td>the IQN target address of iSCSI target in which the SR will reside</td>
<td>yes</td>
</tr>
<tr>
<td>SCSIid</td>
<td>the SCSI bus ID of the destination LUN</td>
<td>yes</td>
</tr>
<tr>
<td>chapuser</td>
<td>the username to be used during CHAP authentication</td>
<td>no</td>
</tr>
<tr>
<td>chappassword</td>
<td>the password to be used during CHAP authentication</td>
<td>no</td>
</tr>
<tr>
<td>port</td>
<td>the network port number on which to query the target</td>
<td>no</td>
</tr>
<tr>
<td>usediscoverynumber</td>
<td>the specific iscsi record index to use</td>
<td>no</td>
</tr>
</tbody>
</table>

To create a shared lvmoiscsi SR on a specific LUN of an iSCSI target use the following command.

```
x e sr-create host-uuid= <valid_uuid> content-type= user 
name-label= "Example shared LVM over iSCSI SR" shared=true 
device-config:target= <target_ip= > device-config:targetIQN= <target_iqn= > 
device-config:SCSIid= <scsi_id= > 
type= lvmoiscsi
```

Creating a shared LVM over Fibre Channel or iSCSI HBA SR (lvmohba)

SRs of type lvmohba can only be created via the xe Command Line Interface (CLI). Once created lvmohba SRs can be managed using either XenCenter or the xe CLI.
Device-config parameters for lvmohba SRs:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCSIId</td>
<td>Device SCSI ID</td>
<td>Yes</td>
</tr>
</tbody>
</table>

To create a shared lvmohba SR, perform the following steps on each host in the pool:

1. Zone in one or more LUNs to each XenServer host in the pool. This process is highly specific to the SAN equipment in use. Please refer to the documentation for your SAN or contact your storage administrator for details.

2. If necessary, use the HBA Command Line Interfaces (CLIs) included in the XenServer host to configure the HBA:
   - Emulex: /usr/sbin/hbanyware
   - QLogic FC: /opt/QLogic_Corporation/SANsurferCLI
   - QLogic iSCSI: /opt/QLogic_Corporation/SANsurferiCLI

   See the section called “Managing Host Bus Adapters (HBAs)” for an example of QLogic iSCSI HBA configuration. For more information on Fibre Channel and iSCSI HBAs please refer to the Emulex and QLogic websites.

3. Use the sr-probe command to determine the global device path of the HBA LUN. sr-probe will force a re-scan of HBAs installed in the system to detect any new LUNs that have been zoned to the host and return a list of properties for each LUN found. Use the host-uuid parameter to ensure the probe occurs on the desired host.

   The global device path returned as the <path> property will be common across all hosts in the pool and therefore must be used as the value for the device-config:device parameter when creating the SR.

   If multiple LUNs are present use the vendor, LUN size, LUN serial number, or the SCSI ID as included in the <path> property to identify the desired LUN.
xe sr-probe type=lvmohba \
host-uuid=1212c7b3-f333-4a8d-a6fb-80c5b79b5b31
Error code: SR_BACKEND_FAILURE_90
Error parameters: , The request is missing the device parameter, \
<?xml version="1.0" ?>
<Devlist>
  <BlockDevice>
    <path>
      /dev/disk/by-id/scsi-360a9800068666949673446387665336f
    </path>
    <vendor>
      HITACHI
    </vendor>
    <serial>
      730157980002
    </serial>
    <size>
      80530636800
    </size>
    <adapter>
      4
    </adapter>
    <channel>
      0
    </channel>
    <id>
      4
    </id>
    <lun>
      2
    </lun>
    <hba>
      qla2xxx
    </hba>
  </BlockDevice>
  <Adapter>
    <host>
      Host4
    </host>
    <name>
      qla2xxx
    </name>
    <manufacturer>
      QLogic HBA Driver
    </manufacturer>
    <id>
      4
    </id>
  </Adapter>
</Devlist>

4. On the master host of the pool create the SR, specifying the global device path returned in the <path> property from sr-probe. PBDs will be created and plugged for each host in the pool automatically.

```
xe sr-create host-uuid=<valid_uuid> \ 
content-type=user \ 
name-label="Example shared LVM over HBA SR" shared=true \ 
device-config:SCSIid=<device_scsi_id> type=lvmohba
```
Note

The Repair Storage Repository function within XenCenter can be used to retry the PBD creation and plugging portions of the sr-create operation. This can be valuable in cases where the LUN zoning was incorrect for one or more member servers in a pool when the SR was created. Correct the zoning for the affected hosts and use Repair Storage Repository instead of removing and re-creating the SR.

Creating a shared NetApp SR over iSCSI

Device-config parameters for netapp SRs:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Optional?</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>the IP address or hostname of the NetApp server in which the SR will reside</td>
<td>no</td>
</tr>
<tr>
<td>username</td>
<td>the login username used to manipulate the LUNs on the filer</td>
<td>no</td>
</tr>
<tr>
<td>password</td>
<td>the login password used to manipulate the LUNs on the filer</td>
<td>no</td>
</tr>
<tr>
<td>aggregate</td>
<td>the aggregate name on which the FlexVol is created</td>
<td>Required for sr_create</td>
</tr>
<tr>
<td>FlexVols</td>
<td>the number of FlexVols to allocate per-SR</td>
<td>yes</td>
</tr>
<tr>
<td>chapuser</td>
<td>the username to be used during CHAP authentication</td>
<td>yes</td>
</tr>
<tr>
<td>chappassword</td>
<td>the password to be used during CHAP authentication</td>
<td>yes</td>
</tr>
<tr>
<td>allocation</td>
<td>this specifies whether thick or thin provisioning [thick</td>
<td>thin]; default is thick</td>
</tr>
<tr>
<td>asis</td>
<td>this specifies whether to use FAS Deduplication if available [true</td>
<td>false]; default is false</td>
</tr>
</tbody>
</table>

Setting the SR other-config:multiplier parameter to a valid value will adjust the default multiplier attribute. By default we allocate 2.4 times the requested space to account for snapshot and metadata overhead associated with each LUN. Note that a valid value means “a value that is supported by the array.” If you try to set the amount to 0.1 and attempt to create a really small VDI, it will likely fail.

Setting the SR other-config:enforce_allocation parameter to true will resize the FlexVols to precisely the amount specified by either the multiplier value above, or the default 2.4 value. Note that this works on new VDI creation in the selected FlexVol, or on all FlexVols during an SR scan. Note also that this will override any manual size adjustments made by the administrator to the SR FlexVols.

To create a NetApp SR, use the following command.

```plaintext
xe sr-create host-uuid=<valid_uuid> content-type=user \
    name-label="Example shared NetApp SR" shared=true \
    device-config:target=192.168.1.10 device-config:username=<admin_username> \
    device-config:password=<admin_password> \
    type=netapp
```
Creating a shared EqualLogic SR

Device-config parameters for EqualLogic SRs:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Optional?</th>
</tr>
</thead>
<tbody>
<tr>
<td>target</td>
<td>the IP address or hostname of the EqualLogic array in which the SR will reside</td>
<td>no</td>
</tr>
<tr>
<td>username</td>
<td>the login username used to manipulate the LUNs on the array</td>
<td>no</td>
</tr>
<tr>
<td>password</td>
<td>the login password used to manipulate the LUNs on the array</td>
<td>no</td>
</tr>
<tr>
<td>storagepool</td>
<td>the storage pool name</td>
<td>no</td>
</tr>
<tr>
<td>chapuser</td>
<td>the username to be used during CHAP authentication</td>
<td>yes</td>
</tr>
<tr>
<td>chappassword</td>
<td>the password to be used during CHAP authentication</td>
<td>yes</td>
</tr>
<tr>
<td>allocation</td>
<td>specifies whether to use thick or thin provisioning [thick</td>
<td>thin]; default is thick. Thin provisioning reserves a minimum of 10% of volume space.</td>
</tr>
<tr>
<td>snap-reserve-percentage</td>
<td>sets the amount of space, as percentage of volume reserve, to allocate to snapshots. Default is 100%.</td>
<td>yes</td>
</tr>
<tr>
<td>snap-depletion</td>
<td>sets the action to take when snapshot reserve space is exceeded. volume-offline sets the volume and all its snapshots offline. This is the default action. The delete-oldest deletes the oldest snapshot until enough space is available for creating the new snapshot.</td>
<td>yes</td>
</tr>
</tbody>
</table>

To create a EqualLogic SR, use the following command.

```
xe sr-create host-uuid=<valid_uuid> content-type=user \
name-label="Example shared Equallogic SR" \
shared=true device-config:target=<target_ip> \
device-config:username=<admin_username> \
device-config:password=<admin_password> \
device-config:storagepool=<my_storagepool> \
device-config:chapuser=<chapusername> \
device-config:chappassword=<chapuserpassword> \
device-config:allocation=<thick> \ntype=equal
```

**Warning**

There are two types of administration accounts that can successfully access the EqualLogic SM plugin:

- A group administration account which has access to and can manage the entire group and all storage pools.
• A pool administrator account that can manage only the objects (SRs and VDIs/Snapshots) that are in the pool or pools assigned to the account.

Storage Multipathing

Dynamic multipathing support is available for Fibre Channel and iSCSI storage backends. By default, it uses round-robin mode load balancing, so both routes will have active traffic on them during normal operation. Multipathing can be enabled in XenCenter or on the command line.

Caution

Before attempting to enable multipathing, verify that multiple targets are available on your storage server. For example, an iSCSI storage backend queried for sendtargets on a given portal should return multiple targets, as in the following example:

```
iscsiadm -m discovery --type sendtargets --portal 192.168.0.161
192.168.0.161:3260,1 iqn.strawberry:litchie
192.168.0.204:3260,2 iqn.strawberry:litchie
```

To enable storage multipathing using the xe CLI

1. Unplug all PBDs on the host:

   ```
   xe pbd-unplug uuid=<pbd_uuid>
   ```

2. Set the host's other-config:multipathing parameter:

   ```
   xe host-param-set other-config:multipathing=true uuid=host_uuid
   ```

3. Set the host's other-config:multipathhandle parameter to dmp:

   ```
   xe host-param-set other-config:multipathhandle=dmp uuid=host_uuid
   ```

4. If there are existing SRs on the host running in single path mode but that have multiple paths:

   • Migrate or suspend any running guests with virtual disks in affected the SRs
   • Unplug and re-plug the PBD of any affected SRs to reconnect them using multipathing:

   ```
   xe pbd-plug uuid=<pbd_uuid>
   ```

To disable multipathing, first unplug your VBDs, set the host other-config:multipathing to false and then replug your PBDs as described above. Do not modify the other-config:multipathhandle as this will be done automatically.
Note

Multipath support in Equallogic arrays does not encompass Storage IO multipathing in the traditional sense of the term. Multipathing must be handled at the network/NIC bond level. Refer to the Equallogic documentation for information about configuring network failover for Equallogic SRs/LVMiSCSI SRs.

Managing Storage Repositories

This section covers various operations required in the ongoing management of Storage Repositories (SRs).

Destroying or forgetting a SR

You can destroy an SR, which will actually delete the contents of the SR from the physical media. Alternatively an SR can be forgotten, which allows you to re-attach the SR, for example, to another XenServer host, without removing any of the SR contents. In both cases, the PBD of the SR must first be unplugged. Forgetting an SR is the equivalent of the SR Detach operation within XenCenter.

1. Unplug the PBD to detach the SR from the corresponding XenServer host:

   ```
   xe pbd-unplug uuid=<pbd_uuid>
   ```

2. To destroy the SR, which deletes both the SR and corresponding PBD from the XenServer host database and deletes the SR contents from the physical media:

   ```
   xe sr-destroy uuid=<sr_uuid>
   ```

3. Or, to forget the SR, which removes the SR and corresponding PBD from the XenServer host database but leaves the actual SR contents intact on the physical media:

   ```
   xe sr-forget uuid=<sr_uuid>
   ```

Introducing an SR

Introducing an SR that has been forgotten requires introducing an SR, creating a PBD, and manually plugging the PBD to the appropriate XenServer hosts in order to activate the SR.

The following example introduces a SR of type lvmoiscsi.

1. Probe the existing SR to determine its UUID:

   ```
   xe sr-probe type=lvmoiscsi device-config:target=192.168.1.10 \
   device-config:targetIQN=192.168.1.10:filer1 \
   device-config:SCSIid=14945540000000000000000002000000b70200000f000000
   ```

2. Introduce the existing SR UUID returned from sr-probe. The UUID for the new SR will be returned:

   ```
   xe sr-introduce content-type=user name-label="Example Shared LVM over iSCSI SR" shared=true uuid=<valid_sr_uuid> type=lvmoiscsi
   ```

3. Create a PBD to accompany the SR. The UUID of the new PBD will be returned:

   ```
   xe pbd-create type=lvmoiscsi host-uuid=<valid_uuid> sr-uuid=<valid_sr_uuid> \
   device-config:target=192.168.0.1 \
   device-config:targetIQN=192.168.1.10:filer1 \
   device-config:SCSIid=14945540000000000000000002000000b70200000f000000
   ```
4. Plug the PBD to attach the SR:

```
xen pbd-plug uuid=<pbd_uuid>
```

5. Verify the status of the PBD plug. If successful the `currently-attached` property will be true:

```
xen pbd-list sr-uuid=<sr_uuid>
```

**Note**

Steps 3 through 5 must be performed for each host in the resource pool, and can also be performed using the Repair Storage Repository function in XenCenter.

### Converting local Fibre Channel SRs to shared SRs

The XenServer 4.0.1 release supported only local (non-shared) Fibre Channel (FC) SRs. In cases where a local FC SR is actually accessible by other hosts in a pool the SR can be converted to shared, allowing VMs with VDIs on the SR to be started on and migrated between hosts within the pool.

Converting a local FC SR to a shared FC SR requires using the `xe` CLI and the XenCenter Repair Storage Repository feature:

1. Upgrade all hosts in the resource pool to XenServer 5.0.0.
2. Ensure all hosts in the pool have the SR's LUN zoned appropriately. See the section called “Probing an SR” for details on using `sr-probe` to verify the LUN is present on each host.
3. Convert the SR to shared:

```
xen sr-param-set shared=true uuid=<local_fc_sr>
```

4. Within XenCenter the SR will be move from the host level to the pool level, indicating it is now shared. The SR will be marked with a red ! to indicate it is not currently plugged on all hosts in the pool.
5. Select the SR and then select the Storage ... Repair Storage Repository menu option.
6. Click Repair to create and plug a PBD for each host in the pool.

### Moving Virtual Disk Images (VDIs) between SRs

The set of VDIs associated with a VM can be copied from one SR to another to accommodate maintenance requirements or tiered storage configurations. XenCenter provides the ability to copy a VM and all of its VDIs to the same or a different SR, and a combination of XenCenter and the `xe` CLI can be used to copy individual VDIs.

#### Copying all of a VM's VDIs to a different SR

The XenCenter Copy VM function will create copies of all VDIs for a selected VM on the same or a different SR. The source VM and VDIs are not affected by default. To move the VM to the selected SR rather than creating a copy, select the "Remove original VM" option in the Copy Virtual Machine dialog box.

1. Shutdown the VM.
2. Within XenCenter select the VM and then select the **VM > Copy VM** menu option.
3. Select the desired target SR.
Copying individual VDIs to a different SR

A combination of the xe CLI and XenCenter can be used to copy individual VDIs between SRs.

1. Shutdown the VM.
2. Use the xe CLI to identify the VDI UUIDs for the VDIs to be moved. If the VM has a DVD drive its `vdi-uuid` will be listed as `not in database` and can be ignored.

```
xe vbd-list vm-uuid=<valid_vm_uuid>
```

**Note**

The `vbd-list` command will display both the VBD and VDI UUIDs. Be sure to record the VDI UUIDs rather than the VBD UUIDs.

3. Within XenCenter select the VM's storage tab. For each VDI to be moved, select the VDI and click on the Detach button. This step can also be done using the CLI command `vbd-destroy`.

**Note**

If you use the `vbd-destroy` command to detach the VDI UUIDs, be sure to first check if the VBD has the parameter `other-config:owner` set to `true`. If so, set it to `false`. Issuing the `vbd-destroy` command with `other-config:owner=true` will also destroy the associated VDI.

4. Use the `vdi-copy` command to copy each of the VM's VDIs to be moved to the desired SR.

```
xe vdi-copy uuid=<valid_vdi_uuid> sr-uuid=<valid_sr_uuid>
```

5. Within XenCenter select the VM's storage tab. Use the Attach button and select the VDIs from the new SR. This step can also be done use the `vbd-create` CLI command.

6. To delete the original VDIs, within XenCenter select the Storage tab of the original SR. The original VDIs will be listed with an empty value for the VM field and can be deleted with the Delete button.

Managing VDIs in a NetApp SR

Due to the complex nature of mapping VM storage objects onto NetApp storage objects such as LUNs, FlexVols and disk Aggregates, the plugin driver makes some general assumptions about how storage objects should be organized. The default number of FlexVols that are managed by an SR instance is 8, named `XenStorage_<SR_UUID>_FV#` where # is a value between 0 and the total number of FlexVols assigned. This means that VDIs (LUNs) are evenly distributed across any one of the FlexVols at the point that the VDI is instantiated. The only exception to this rule is for groups of VM disks which are opportunistically assigned to the same FlexVol to assist with VM cloning, and when VDIs are created manually but passed a 'vmhint' flag which informs the backend of the FlexVol to which the VDI should be assigned. Using either of the following 2 commands, a VDI created manually via the CLI can be assigned to a specific FlexVol:

```
xe vdi-create uuid=<valid_vdi_uuid> sr-uuid=<valid_sr_uuid> \
sm-config:vmhint=<valid_vm_uuid>
```

```
xe vdi-create uuid=<valid_vdi_uuid> sr-uuid=<valid_sr_uuid> \
sm-config:vmhint=<valid_flexvol_number>
```
Taking VDI snapshots with a NetApp SR

As outlined earlier in the section called “Shared NetApp Storage”, a NetApp SR comprises a collection of FlexVols. Cloning a VDI entails generating a snapshot of the FlexVol and then creating a LUN clone backed off the snapshot. When generating a VM snapshot, an admin must snapshot each of the VMs disks in sequence. Since all the disks are expected to be located in the same FlexVol, and the FlexVol snapshot operates on all LUNs in the same FlexVol, it makes sense to re-use an existing snapshot for all subsequent LUN clones. By default, if no snapshot hint is passed into the backend driver it will generate a random ID with which to name the FlexVol snapshot. There is a CLI override however for this value, passed in as an epochhint. The first time the epochhint value or 'cookie' is received, the backend will generate a new snapshot based on the cookie name. Any subsequent snapshot requests with the same epochhint value will be backed off the existing snapshot:

```
xe vdi-snapshot uuid=<valid_vdi_uuid> driver-params:epochhint=<cookie>
```

During provisioning of a NetApp SR, additional disk space is reserved for snapshots. If you never plan to use the snapshotting functionality, you might want to free up this reserved space. To do so, you can reduce the value of the other-config:multiplier parameter. By default the value of the multiplier is 2.4, so the amount of space reserved is 2.4 times the amount of space that would be needed for the FlexVols themselves.

Adjusting the disk IO scheduler for an LVM-based SR

For general performance, the default disk scheduler noop is applied on all new SR types that implement LVM based storage over a disk, i.e. Local LVM, LVM over iSCSI and LVM over HBA attached LUNs. The noop scheduler provides the fairest performance for competing VMs accessing the same device. In order to apply disk QoS however (the section called “Virtual disk QoS settings (Enterprise Edition only)”) it is necessary to override the default setting and assign the 'cfq' disk scheduler to any LVM-based SR type. For any LVM-based SR type, the corresponding PBD must be unplugged and re-plugged in order for the scheduler parameter to take effect. The disk scheduler can be adjusted using the following CLI parameter:

```
xe sr-param-set other-config:scheduler={noop|cfq|anticipatory|deadline} uuid=<valid_sr_uuid>
```

Managing Host Bus Adapters (HBAs)

This section covers various operations required to manage Fibre Channel and iSCSI HBAs.

Sample QLogic iSCSI HBA setup

For full details on configuring QLogic Fibre Channel and iSCSI HBAs please refer to the QLogic website.

Once the HBA is physically installed into the XenServer host, use the following steps to configure the HBA:

1. Set the IP networking configuration for the HBA. This example assumes DHCP and HBA port 0. Specify the appropriate values if using static IP addressing or a multi-port HBA.

```
/opt/QLogic_Corporation/SANsurferiCLI/iscli -ipdhcp 0
```

2. Add a persistent iSCSI target to port 0 of the HBA.

```
/opt/QLogic_Corporation/SANsurferiCLI/iscli -pa 0 <iscsi_target_ip_address> -INAME <iscsi_target_iqn>
```

3. Use the xe sr-probe command to force a rescan of the HBA controller and display available LUNs. See the section called “Probing an SR” and the section called “Creating a shared LVM over Fibre Channel or iSCSI HBA SR (lvmohba)” for more details.
Removing HBA-based FC or iSCSI device entries

**Note**

This step is not required. Citrix recommends that only power users perform this process if it is necessary.

Each HBA-based LUN has a corresponding global device path entry under `/dev/disk/by-id` and a standard device path under `/dev`. To remove the device entries for LUNs no longer in use as SRs use the following steps:

1. Use `sr-forget` or `sr-destroy` as appropriate to remove the SR from the XenServer host database. See the section called “Destroying or forgetting a SR” for details.
2. Remove the zoning configuration within the SAN for the desired LUN to the desired host.
3. Use the `sr-probe` command to determine the ADAPTER, BUS, TARGET, and LUN values corresponding to the LUN to be removed. See the section called “Probing an SR” for details.
4. Remove the device entries with the following command:

   ```bash
   echo "1" > /sys/class/scsi_device/<adapter>:<bus>:<target>:<lun>/device/delete
   ```

**Warning**

Make absolutely sure you are certain which LUN you are removing. Accidentally removing a LUN required for host operation, such as the boot or root device, will render the host unusable.

Enabling Virtual HBA support

Some storage devices require the use of the `mppVhba` driver, which is not enabled by default.

**To enable the mppVhba driver**

1. Edit the `/etc/mpp.conf` file on your XenServer host as described in your storage product documentation.
2. Run the `/opt/xensource/bin/update-initrd` script to rebuild the `initrd`. This script ensures that the correct files are loaded before boot to support the configuration in your `/etc/mpp.conf` file, and support the `mppVhba` driver.
3. Reboot your host.

**Note**

XenServer does not support using an `mppVhba` disk for the root filesystem of a control domain.

Virtual disk QoS settings (Enterprise Edition only)

In Enterprise Edition, virtual disks have an optional I/O priority Quality of Service (QoS) setting. This setting can be made to existing virtual disks with the CLI as described in this section.
In order to enable QoS for disk I/O the SR type underlying the VDI must be an LVM-based volume. QoS will only take effect therefore on SRs of type Local LVM, LVM over iSCSI and LVM over HBA attached LUNs. Note also that in the shared SR case i.e. where multiple hosts are accessing the same LUN, the QoS is applied to VBDs accessing the LUN from the same host, i.e. QoS is not applied across hosts in the pool. Note that QoS settings will not have any effect on VHD-based storage types.

Before configuring any QoS parameters for a VBD, ensure that the disk scheduler for the SR has been set appropriately. See the section called “Adjusting the disk IO scheduler for an LVM-based SR” for details on how to adjust the scheduler. The scheduler parameter must be set to cfq on the SR for which the QoS is desired.

**Note**

Don't forget to set the scheduler to 'cfq' on the SR, and make sure that the PBD has been re-plugged in order for the scheduler change to take effect.

The first parameter is `qos_algorithm_type`. This parameter needs to be set to the value `ionice`, which is the only type of QoS algorithm supported for virtual disks in this release.

The QoS parameters themselves are set with key/value pairs assigned to the `qos_algorithm_type` parameter. For virtual disks, `qos_algorithm_type` takes a sched key, and depending on the value, also requires a class key.

Possible values of `qos_algorithm_type:sched` are

- `sched=rt` or `sched=real-time` sets the QoS scheduling parameter to real time priority, which requires a class parameter to set a value
- `sched=idle` sets the QoS scheduling parameter to idle priority, which requires no class parameter to set any value
- `sched=<anything>` sets the QoS scheduling parameter to best effort priority, which requires a class parameter to set a value

The possible values for class are

- One of the following keywords: highest, high, normal, low, lowest
- an integer between 0 and 7, where 0 is the highest priority and 7 is the lowest

To enable the disk QoS settings, you also need to set the `other-config:scheduler` to `cfq` and replug PBDs for the storage in question.

For example, the following CLI commands set the virtual disk's VBD to use real time priority=5:

```bash
xe vbd-param-set uuid=<vbd_uuid> qos_algorithm_type=ionice
xe vbd-param-set uuid=<vbd_uuid> qos_algorithm_params:sched=rt
xe vbd-param-set uuid=<vbd_uuid> qos_algorithm_params:class=5
xe pbd-param-set uuid=<pbd_uuid> other-config:scheduler-cfg
xe pbd-plug uuid=<pbd_uuid>
```
Chapter 4. Networking

This chapter discusses how physical network interface cards (NICs) in XenServer hosts are used to enable networking within Virtual Machines (VMs). XenServer supports up to 6 physical network interfaces (or up to 6 pairs of bonded network interfaces) per XenServer host and up to 7 virtual network interfaces per VM.

XenServer provides automated configuration and management of NICs via the xe command line interface (CLI).

Note

Unlike previous XenServer versions, the host's networking configuration files should not be edited directly in most cases; where a CLI command is available, do not edit the underlying files.

Some networking options have different behaviors when used with standalone XenServer hosts compared to resource pools. This chapter contains sections on general information that applies to both standalone hosts and pools, followed by specific information and procedures for each.

XenServer networking concepts

This section describes the general concepts of networking in the XenServer environment.

Network objects

There are three types of server-side software objects which represent networking entities. These objects are:

- A **PIF**, which represents a physical network interface on a XenServer host. PIF objects have a name and description, a globally unique UUID, the parameters of the NIC that they represent, and the network and server they are connected to.
- A **VIF**, which represents a virtual interface on a Virtual Machine. VIF objects have a name and description, a globally unique UUID, and the network and VM they are connected to.
- A **network**, which is a virtual Ethernet switch on a XenServer host. Network objects have a name and description, a globally unique UUID, and the collection of VIFs and PIFs connected to them.

Both XenServer and the xe CLI allow configuration of networking options, control over which NIC is used for management operations, and creation of advanced networking features such as virtual local area networks (VLANs) and NIC bonds.

From XenCenter much of the complexity of XenServer networking is hidden. There is no mention of PIFs for XenServer hosts nor VIFs for VMs.

Networks

Each XenServer host has one or more networks, which are virtual Ethernet switches. Networks without an association to a PIF are considered *internal*, and can be used to provide connectivity only between VMs on a given XenServer host, with no connection to the outside world. Networks with a PIF association are considered *external*, and provide a bridge between VIFs and the PIF connected to the network, enabling connectivity to resources available through the PIF’s NIC.

For procedures on how to create networks for standalone XenServer hosts, see the section called “Creating networks in a standalone server”.
For procedures on how to create networks for XenServer hosts that are configured in a resource pool, see the section called “Creating networks in resource pools”.

**VLANs**

Virtual Local Area Networks (VLANs) allow a single physical network to support multiple logical networks. To use VLANs with XenServer, the host's NIC must be connected to a VLAN trunk port.

XenServer VLANs are represented by additional PIF objects representing VLAN interfaces corresponding to a specified VLAN tag. XenServer networks can then be connected to the PIF representing the physical NIC to see all traffic on the NIC, or to a PIF representing a VLAN to see only the traffic with the specified VLAN tag.

When using VLANs the XenServer host handles all interpretation of the VLAN tags and strips the VLAN tags before routing packets to VMs.

For procedures on how to create VLANs for XenServer hosts, either standalone or part of a resource pool, see the section called “Creating VLANs”.

**NIC bonds**

NIC bonds can improve XenServer host resiliency by using two physical NICs as if they were one. If one NIC within the bond fails the host's network traffic will automatically be routed over the second NIC. NIC bonds work in an active/active mode, with traffic balanced between the bonded NICs.

XenServer NIC bonds completely subsume the underlying physical devices (PIFs). In order to activate a bond the underlying PIFs must not be in use, either as the management interface for the host or by running VMs with VIFs attached to the networks associated with the PIFs.

XenServer NIC bonds are represented by additional PIFs. The bond PIF can then be connected to a XenServer network to allow VM traffic and host management functions to occur over the bonded NIC. The exact steps to use to create a NIC bond depend on the number of NICs in your host, and whether the management interface of the host is assigned to a PIF to be used in the bond.

XenServer supports Source Level Balancing (SLB) NIC bonding. SLB bonding:

- is an active/active mode, but only supports load-balancing of VM traffic across the physical NICs
- provides fail-over support for all other traffic types
- does not require switch support for Etherchannel or 802.3ad (LACP)
- load balances traffic between multiple interfaces at VM granularity by sending traffic through different interfaces based on the source MAC address of the packet
- is derived from the open source ALB mode and reuses the ALB capability to dynamically re-balance load across interfaces

Any given VIF will only utilize one of the links in the bond at a time. At startup no guarantees are made about the affinity of a given VIF to a link in the bond. However, for VIFs with high throughput, periodic rebalancing ensures that the load on the links is approximately equal.

API Management traffic can be assigned to a XenServer 5.0.0 bond interface and will be automatically load-balanced across the physical NICs.

XenServer 5.0.0 bonded PIFs do not require IP configuration for the bond when used for guest traffic. This is because the bond operates at Layer 2 of the OSI, the data link layer, and no IP addressing is used at this layer. When used for non-guest traffic (to connect to it with XenCenter for management, or to connect to shared network storage), one IP configuration is required per bond. (Incidentally, this is true of unbonded PIFs as well, and is unchanged from XenServer 4.1.0.)
As in XenServer 4.1.0, gratuitous ARP packets are sent when assignment of traffic changes from one interface to another as a result of fail-over.

Re-balancing is provided by the existing ALB re-balance capabilities: the number of bytes going over each slave (interface) is tracked over a given period. When a packet is to be sent that contains a new source MAC address it is assigned to the slave interface with the lowest utilization. Traffic is re-balanced every 10 seconds.

**Note**

On upgrade from XenServer 4.1.0 all existing managed NIC bonds will be converted to SLB mode.

For procedures on how to create bonds for standalone XenServer hosts, see the section called “Creating NIC bonds on a standalone host”.

For procedures on how to create bonds for XenServer hosts that are configured in a resource pool, see the section called “Creating NIC bonds in resource pools”.

**Initial networking configuration**

The networking configuration for a XenServer host is specified during initial host installation. Options such as IP address configuration (DHCP/static), the NIC used as the management interface, and hostname are set based on the values provided during installation.

When a XenServer host has a single NIC, the follow configuration will be present after installation:

- a single PIF is created corresponding to the host’s single NIC
- the PIF is configured with the IP addressing options specified during installation and to enable management of the host
- the PIF is set for use in host management operations
- a single network, network 0, is created
- network 0 is connected to the PIF to enable external connectivity to VMs

When a host has multiple NICs the configuration present after installation depends on which NIC is selected for management operations during installation:

- PIFs are created for each NIC in the host
- the PIF of the NIC selected for use as the management interface is configured with the IP addressing options specified during installation
- a network is created for each PIF ("network 0", "network 1", etc.)
- each network is connected to one PIF
- the IP addressing options of all other PIFs are left unconfigured

In both cases the resulting networking configuration allows connection to the XenServer host by XenCenter, the xe CLI, and any other management software running on separate machines via the IP address of the management interface. The configuration also provides external networking for VMs created on the host.

The PIF used for management operations is the only PIF ever configured with an IP address. External networking for VMs is achieved by bridging PIFs to VIFs via the network object which acts as a virtual Ethernet switch.

The steps required for networking features such as VLANs, NIC bonds, and dedicating a NIC to storage traffic are covered in the following sections.
Managing networking configuration

Some of the network configuration procedures in this section differ depending on whether you are configuring a stand-alone server or a server that is part of a resource pool.

Creating networks in a standalone server

Because external networks are created for each PIF during host installation, creating additional networks is typically only required to:

- use an internal network
- support advanced operations such as VLANs or NIC bonding

To add or remove networks using XenCenter, refer to the XenCenter online Help.

To add a new network via the CLI

1. Open the XenServer host text console.
2. Create the network with the network-create command, which returns the UUID of the newly created network:

```
xe network-create name-label=<mynetwork>
```

At this point the network is not connected to a PIF and therefore is internal.

Creating networks in resource pools

All XenServer hosts in a resource pool should have the same number of physical network interface cards (NICs), although this requirement is not strictly enforced when a XenServer host is joined to a pool.

Having the same physical networking configuration for XenServer hosts within a pool is important because all hosts in a pool share a common set of XenServer networks. PIFs on the individual hosts are connected to pool-wide networks based on device name. For example, all XenServer hosts in a pool with an eth0 NIC will have a corresponding PIF plugged into the pool-wide "Network 0" network. The same will be true for hosts with eth1 NICs and "Network 1", as well as other NICs present in at least one XenServer host in the pool.

If one XenServer host has a different number of NICs than other hosts in the pool, complications can arise because not all pool networks will be valid for all pool hosts. For example, if hosts host1 and host2 are in the same pool and host1 has four NICs while host2 only has two, only the networks connected to PIFs corresponding to eth0 and eth1 will be valid on host2. VMs on host1 with VIFs connected to networks corresponding to eth2 and eth3 will not be able to migrate to host host2.

All NICs of all XenServer hosts within a resource pool must be configured with the same MTU size.

Creating VLANs

To connect a network to an external VLAN via the CLI

1. Open the XenServer host text console.
2. Create a new network for use with the VLAN. The UUID of the new network is returned:

```
xe network-create name-label=network5
```

3. Use the `pif-list` command to find the UUID of the PIF corresponding to the physical NIC supporting the desired VLAN tag. The UUIDs and device names of all PIFs are returned, including any existing VLANs:
4. Create a VLAN object specifying the desired physical PIF and VLAN tag on all VMs to be connected to the new VLAN. A new PIF will be created and plugged into the specified network. The UUID of the new PIF object is returned.

   xe vlan-create network-uuid= <network_uuid>  pif-uuid= <pif_uuid>  vlan=5

5. Attach VM VIFs to the new network. See the section called “Creating networks in a standalone server” for more details.

Creating NIC bonds on a standalone host

We recommend using XenCenter to create NIC bonds. For details, refer to the XenCenter help.

This section describes using the xe CLI to create bonded NIC interfaces on a standalone XenServer host. See the section called “Creating NIC bonds in resource pools” for details on using the xe CLI to create NIC bonds on XenServer hosts that comprise a resource pool.

Creating a NIC bond on a dual-NIC host

Creating a bond on a dual-NIC host implies that the PIF/NIC currently in use as the management interface for the host will be subsumed by the bond. The additional steps required to move the management interface to the bond PIF are included.

Bonding two NICs together

1. Use XenCenter or the vm-shutdown command to shut down all VMs on the host, thereby forcing all VIFs to be unplugged from their current networks. The existing VIFs will be invalid after the bond is enabled.

   xe vm-shutdown uuid= <vm_uuid>

2. Use the network-create command to create a new network for use with the bonded NIC. The UUID of the new network is returned:

   xe network-create name-label= <bond0>

3. Use the pif-list command to determine the UUIDs of the PIFs to use in the bond:

   xe pif-list

4. Use the bond-create command to create the bond by specifying the newly created network UUID and the UUIDs of the PIFs to be bonded separated by commas. The UUID for the bond is returned:

   xe bond-create network-uuid= <network_uuid>  pif-uuids= <pif_uuid_1>, <pif_uuid_2>

   Note

   See the section called “Controlling the MAC address of the bond” for details on controlling the MAC address used for the bond PIF.

5. Use the pif-list command to determine the UUID of the new bond PIF:
6. Use the `pif-reconfigure-ip` command to configure the desired management interface IP address settings for the bond PIF. See Chapter 6, *Command line interface* for more detail on the options available for the pif-reconfigure-ip command.

```
xe pif-reconfigure-ip device= <bond0>
```

7. Use the `host-management-reconfigure` command to move the management interface from the existing physical PIF to the bond PIF. This step will activate the bond:

```
xe host-management-reconfigure pif-uuid= <bond_pif_uuid>
```

8. Use the `pif-reconfigure-ip` command to remove the IP address configuration from the non-bonded PIF previously used for the management interface. This step is not strictly necessary but might help reduce confusion when reviewing the host networking configuration.

```
xe pif-reconfigure-ip uuid= <old_management_pif_uuid> mode= None
```

9. Move existing VMs to the bond network using the `vif-destroy` and `vif-create` commands. This step can also be completed via XenCenter by editing the VM configuration and connecting the existing VIFs of a VM to the bond network.

10. Restart the VMs shut down in step 1.

### Controlling the MAC address of the bond

Creating a bond on a dual-NIC host implies that the PIF/NIC currently in use as the management interface for the host will be subsumed by the bond. If DHCP is used to supply IP addresses to the host in most cases the MAC address of the bond should be the same as the PIF/NIC currently in use, allowing the IP address of the host received from DHCP to remain unchanged.

The MAC address of the bond can be changed from PIF/NIC currently in use for the management interface, but doing so will cause existing network sessions to the host to be dropped when the bond is enabled and the MAC/IP address in use changes.

The MAC address to be used for a bond can be controlled in two ways:

- an optional `mac` parameter can be specified in the `bond-create` command. Using this parameter, the bond MAC address can be set to any arbitrary address.
- If the `mac` parameter is not specified, the MAC address of the first PIF listed in the `pif-uuids` parameter is used for the bond.

### Reverting NIC bonds

If reverting a XenServer host to a non-bonded configuration, be aware of the following requirements:

- As when creating a bond, all VMs with VIFs on the bond must be shut down prior to destroying the bond. After reverting to a non-bonded configuration, reconnect the VIFs to an appropriate network.
- Move the management interface to another PIF using the `pif-reconfigure-ip` and `host-management-reconfigure` commands prior to issuing the `bond-destroy` command, otherwise connections to the host (including XenCenter) will be dropped.

### Creating NIC bonds in resource pools

Whenever possible, create NIC bonds as part of initial resource pool creation prior to joining additional member servers to the pool or creating VMs. Doing so allows the bond configuration to be automatically replicated to member servers.
as they are joined to the pool and reduces the number of steps required. Adding a NIC bond to an existing pool requires creating the bond configuration manually on the master and each of the members of the pool. Adding a NIC bond to an existing pool after VMs have been installed is also a disruptive operation, as all VMs in the pool must be shut down.

We recommend using XenCenter to create NIC bonds. For details, refer to the XenCenter help.

This section describes using the xe CLI to create bonded NIC interfaces on XenServer hosts that comprise a resource pool. See the section called “Creating a NIC bond on a dual-NIC host” for details on using the xe CLI to create NIC bonds on a standalone XenServer host.

**Adding NIC bonds to new resource pools**

1. Select the host you want to be the master. The master host belongs to an unnamed pool by default. To create a resource pool with the CLI, rename the existing nameless pool:

   ```bash
   xe pool-param-set name-label="New Pool" uuid=pool_uuid
   ```

2. Create the NIC bond on the master as follows:

   a. Use the `network-create` command to create a new pool-wide network for use with the bonded NICs. The UUID of the new network is returned.

   ```bash
   xe network-create name-label=network_name
   ```

   b. Use the `host-list` command to find the UUID of the master host:

   ```bash
   xe host-list
   ```

   c. Use the `pif-list` command to determine the UUIDs of the PIFs to use in the bond:

   ```bash
   xe pif-list
   ```

   d. Use the `bond-create` command to create the bond, specifying the network UUID created in step a and the UUIDs of the PIFs to be bonded, separated by commas. The UUID for the bond is returned:

   ```bash
   xe bond-create network-uuid=network_uuid pif-uuids=pif_uuid_1,pif_uuid_2
   ```

   Note

   See the section called “Controlling the MAC address of the bond” for details on controlling the MAC address used for the bond PIF.

   e. Use the `pif-list` command to determine the UUID of the new bond PIF:

   ```bash
   xe pif-list device=network_name
   ```

   f. Use the `pif-reconfigure-ip` command to configure the desired management interface IP address settings for the bond PIF. See Chapter 6, Command line interface, for more detail on the options available for the `pif-reconfigure-ip` command.

   ```bash
   xe pif-reconfigure-ip uuid=bond_pif_uuid mode=DHCP
   ```
g. Use the `host-management-reconfigure` command to move the management interface from the existing physical PIF to the bond PIF. This step will activate the bond:

```
xhost-management-reconfigure pif-uuid=<bond_pif_uuid>
```

h. Use the `pif-reconfigure-ip` command to remove the IP address configuration from the non-bonded PIF previously used for the management interface. This step is not strictly necessary but might help reduce confusion when reviewing the host networking configuration.

```
xpif-reconfigure-ip uuid=<old_management_pif_uuid> mode=None
```

3. Open a console on a host that you want to join to the pool and type:

```
x pool-join master-address=<host1> master-username=root master-password=<password>
```

The network and bond information will be automatically replicated to the member server. However, the management interface is not automatically moved from the member server's NIC to the bonded NIC. Move the management interface on the member server to enable the bond as follows:

a. Use the `host-list` command to find the UUID of the member host being configured:

```
x host-list
```

b. Use the `pif-list` command to determine the UUID of bond PIF on the new member host. Include the `host-uuid` parameter to list only the PIFs on the host being configured:

```
xpif-list device=<network_name> host-uuid=<host_uuid>
```

c. Use the `pif-reconfigure-ip` command to configure the desired management interface IP address settings for the bond PIF with the `pif-reconfigure-ip` command. See Chapter 6, *Command line interface*, for more detail on the options available for the `pif-reconfigure-ip` command. *This command must be run directly on the member server:*

```
xpif-reconfigure-ip uuid=<bond_pif_uuid> mode=DHCP
```

d. Use the `host-management-reconfigure` command to move the management interface from the existing physical PIF to the bond PIF. This step will activate the bond. *This command must be run directly on the member server:*

```
x host-management-reconfigure pif-uuid=<bond_pif_uuid>
```

e. Use the `pif-reconfigure-ip` command to remove the IP address configuration from the non-bonded PIF previously used for the management interface. This step is not strictly necessary but may help reduce confusion when reviewing the host networking configuration. *This command must be run directly on the member server:*

```
xpif-reconfigure-ip uuid=<old_mgmt_pif_uuid> mode=None
```

4. For each additional member server you want to join to the pool, repeat steps 3 and 4 to move the management interface on the member server to enable the bond.

**Adding NIC bonds to an existing pool**

When adding a NIC bond to an existing pool, the bond must be manually created on each host in the pool. The steps below can be used to add NIC bonds on both the pool master and member servers with the following requirements:
1. All VMs in the pool must be shut down
2. Add the bond to the pool master first, and then to member hosts.
3. The **bond-create**, **host-management-reconfigure** and **host-management-disable** commands affect the host on which they are run and so are not suitable for use on one host in a pool to change the configuration of another. Run these commands directly on the console of the host to be affected.

To add NIC bonds to existing pool master and member hosts

1. Use the **network-create** command to create a new pool-wide network for use with the bonded NICs. This step should only be performed once per pool. The UUID of the new network is returned.

```bash
xe network-create name-label=<bond0>
```

2. Use XenCenter or the **vm-shutdown** command to shut down all VMs in the host pool to force all existing VIFs to be unplugged from their current networks. The existing VIFs will be invalid after the bond is enabled.

```bash
xe vm-shutdown uuid=<vm_uuid>
```

3. Use the **host-list** command to find the UUID of the host being configured:

```bash
xe host-list
```

4. Use the **pif-list** command to determine the UUIDs of the PIFs to use in the bond. Include the **host-uuid** parameter to list only the PIFs on the host being configured:

```bash
xe pif-list host-uuid=<host_uuid>
```

5. Use the **bond-create** command to create the bond, specifying the network UUID created in step 1 and the UUIDs of the PIFs to be bonded, separated by commas. The UUID for the bond is returned.

```bash
xe bond-create network-uuid=<network_uuid> pif-uuids=<pif_uuid_1>,<pif_uuid_2>
```

**Note**

See the section called “Controlling the MAC address of the bond” for details on controlling the MAC address used for the bond PIF.

6. Use the **pif-list** command to determine the UUID of the new bond PIF. Include the **host-uuid** parameter to list only the PIFs on the host being configured:

```bash
xe pif-list device=bond0 host-uuid=<host_uuid>
```

7. Use the **pif-reconfigure-ip** command to configure the desired management interface IP address settings for the bond PIF. See Chapter 6, **Command line interface** for more detail on the options available for the **pif-reconfigure-ip** command. *This command must be run directly on the member server*:

```bash
xe pif-reconfigure-ip uuid=<bond_pif_uuid> mode=DHCP
```

8. Use the **host-management-reconfigure** command to move the management interface from the existing physical PIF to the bond PIF. This step will activate the bond. *This command must be run directly on the member server*:

```bash
xe host-management-reconfigure pif-uuid=<bond_pif_uuid>
```
9. Use the `pif-reconfigure-ip` command to remove the IP address configuration from the non-bonded PIF previously used for the management interface. This step is not strictly necessary, but might help reduce confusion when reviewing the host networking configuration. This command must be run directly on the member server:

```
xe pif-reconfigure-ip uuid=<old_management_pif_uuid> mode=None
```

10. Move existing VMs to the bond network using the `vif-destroy` and `vif-create` commands. This step can also be completed via XenCenter by editing the VM configuration and connecting the existing VIFs of the VM to the bond network.

11. Repeat steps 3 - 10 for member servers.

12. Restart the VMs previously shut down.

### Configuring a dedicated storage NIC

XenServer 4.1 allowed dedicating NICs to storage traffic by configuring the NIC as unmanaged with the `xe CLI and requiring manual configuration of the underlying networking settings for that NIC within the control domain. XenServer 5.0.0 allows use of either XenCenter or the `xe CLI to configure and dedicate a NIC to specific functions, such as storage traffic.

Assigning a NIC to a specific function will prevent the use of the NIC for other functions such as host management, but requires that the appropriate network configuration be in place in order to ensure the NIC is used for the desired traffic. For example, to dedicate a NIC to storage traffic the NIC, storage target, switch, and/or VLAN must be configured such that the target is only accessible over the assigned NIC. This allows use of standard IP routing to control how traffic is routed between multiple NICs within a XenServer.

**Note**

Before dedicating a network interface as a storage interface for use with iSCSI or NFS SRs, you must ensure that the dedicated interface uses a separate IP subnet which is not routable from the main management interface. If this is not enforced, then storage traffic may be directed via the main management interface after a host reboot, due to the order in which network interfaces are initialized.

**To assign NIC functions using the `xe CLI**

1. Ensure the PIF is on a separate subnet, or routing is configured to suit your network topology in order to force the desired traffic over the selected PIF.

2. Setup an IP configuration for the PIF, adding appropriate values for the mode parameter and if using static IP addressing the IP, netmask, gateway, and DNS parameters:

   ```
   xe pif-reconfigure-ip mode=<DHCP | Static> uuid=<pif-uuid>
   ```

3. Set the PIF's disallow-unplug parameter to true:

   ```
   xe pif-param-set disallow-unplug=true uuid=<pif-uuid>
   ```

If you do wish to use a storage interface which can be routed from the management interface also (bearing in mind that this configuration is not recommended), then you have two options:
• After a host reboot, ensure that the storage interface is correctly configured, and use the `xe pbd-unplug` and `xe pbd-plug` commands to reinitialize the storage connections on the host. This will restart the storage connection and route it over the correct interface.

• Alternatively, you can use `xe pif-forget` to remove the interface from the XenServer database, and manually configure it in the control domain. This is an advanced option and requires you to be familiar with how to manually configure Linux networking.

Controlling Quality of Service (QoS)

Enterprise Edition allows an optional Quality of Service (QoS) value to be set on VM virtual network interfaces (VIFs) using the CLI. The only QoS algorithm type supported in this release is rate limiting, specified as a maximum transfer rate for the VIF in Kb/s.

For example, to limit a VIF to a maximum transfer rate of 100kb/s, use the `vif-param-set` command:

```shell
xe vif-param-set uuid=<vif_uuid> qos_algorithm_type=ratelimit
xe vif-param-set uuid=<vif_uuid> qos_algorithm_params:kbps=100
```

Changing networking configuration options

This section discusses how to change the networking configuration of a XenServer host. This includes:

• changing the hostname
• adding or removing DNS servers
• changing IP addresses
• changing which NIC is used as the management interface
• adding a new physical NIC to the server

Hostname

The system host-name is defined in the pool-wide database and modified using the `xe host-set-hostname-live` CLI command as follows:

```shell
xe host-set-hostname-live uuid=<host_uuid> host-name=example
```

The underlying control domain hostname will also change dynamically to reflect the new hostname.

DNS servers

To add or remove DNS servers in the IP addressing configuration of a XenServer host, use the `pif-reconfigure-ip` command. For example, for a PIF with a static IP:

```shell
pif-reconfigure-ip uuid=<pif_uuid> mode=static DNS=<new_dns_ip>
```

Changing IP address configuration for a standalone host

Network interface configuration can be manipulated via the `xe` command-line interface (CLI). Unlike earlier XenServer versions, the underlying network configuration scripts should not be modified directly without using the CLI.

To modify the IP address configuration of a PIF, use the `pif-reconfigure-ip` CLI command. See the section called “pif-reconfigure-ip” for details on the parameters of the `pif-reconfigure-ip` command.
Note
See the section called “Changing IP address configuration in resource pools” for details on changing host IP addresses in resource pools.

Changing IP address configuration in resource pools

XenServer hosts in resource pools have a single management IP address used for management and communication to and from other hosts in the pool. The steps required to change the IP address of a host's management interface are different for master and member hosts.

Note
Caution should be used when changing the IP address of a server, and other networking parameters. Depending upon the network topology and the change being made, connections to network storage may be lost. If this happens the storage must be replugged using the Repair Storage option in the user interface, or the `pbd-plug` command via the CLI. For this reason, it may be advisable to migrate VMs away from the server before changing its IP configuration.

Changing the IP address of a pool member host

1. Use the `pif-reconfigure-ip` CLI command to set the IP address as desired. See Chapter 6, Command line interface for details on the parameters of the `pif-reconfigure-ip` command:

   ```bash
   xe pif-reconfigure-ip uuid=<pif_uuid> mode=DHCP
   ```

2. Use the `host-list` CLI command to confirm that the member host has successfully reconnected to the master host by checking that all the other XenServer hosts in the pool are visible:

   ```bash
   xe host-list
   ```

Changing the IP address of the master XenServer host requires additional steps because each of the member hosts uses the master's advertised IP address for communication and will not know how to contact the master when its IP address changes.

Whenever possible, use a dedicated IP address that is not likely to change for the lifetime of the pool for pool masters.

To change the IP address of a pool master host

1. Use the `pif-reconfigure-ip` CLI command to set the IP address as desired. See Chapter 6, Command line interface for details on the parameters of the `pif-reconfigure-ip` command:

   ```bash
   xe pif-reconfigure-ip uuid=<pif_uuid> mode=DHCP
   ```

2. When the IP address of the pool master host is changed, all member hosts will enter into an emergency mode when they fail to contact the master host.

3. On the master XenServer host, use the `pool-recover-slaves` command to force the master to contact each of the member servers and inform them of the master's new IP address:

   ```bash
   xe pool-recover-slaves
   ```
Refer to the section called “Master failures” for more information on emergency mode.

**Management interface**

When XenServer is installed on a host with multiple NICs, one NIC is selected for use as the management interface. The management interface is used for XenCenter connections to the host and for host-to-host communication.

**To change the NIC used for the management interface**

1. Use the `pif-list` command to determine which PIF corresponds to the NIC desired for use as the management interface. The UUID of each PIF will be returned.

   xe pif-list

2. Use the `pif-param-list` command to verify the IP addressing configuration for the PIF that will be used for the management interface. If necessary, use the `pif-reconfigure-ip` command to configure IP addressing for the PIF to be used. See Chapter 6, Command line interface for more detail on the options available for the `pif-reconfigure-ip` command.

   xe pif-param-list uuid=<pif_uuid>

3. Use the `host-management-reconfigure` CLI command to change the PIF used for the management interface. If this host is part of a resource pool, this command must be issued on the member host console:

   xe host-management-reconfigure pif-uuid=<pif_uuid>

**Warning**

Putting the management interface on a VLAN network is not supported.

**Disabling management access**

To disable remote access to the management console entirely, use the `host-management-disable` CLI command. But be careful! Once the management interface is disabled, you will have to log in on the physical host console to perform management tasks and external interfaces such as XenCenter will no longer work.

**Adding a new physical NIC**

Install a new physical NIC on a XenServer host in the usual manner. Then, after restarting the server, run the `xe` CLI command `pif-scan` to cause a new PIF object to be created for the new NIC.

**NIC/PIF ordering in resource pools**

It is possible for physical NIC devices to be discovered in different orders on different servers even though the servers contain the same hardware. Verifying NIC ordering is recommended before using the pooling features of XenServer.

**Verifying NIC ordering**

The `pif-list` command can be used to verify that NIC ordering is consistent across your XenServer hosts. Review the MAC address and carrier (link state) parameters associated with each PIF to verify that the devices discovered (eth0, eth1, etc.) correspond to the appropriate physical port on the server.
If the hosts have already been joined in a pool, add the `host-uuid` parameter to the `pif-list` command to scope the results to the PIFs on a given host.

**Re-ordering NICs**

It is not possible to directly rename a PIF, although you can use the `pif-forget` and `pif-introduce` commands to achieve the same effect with the following restrictions:

- The XenServer host must be standalone and not joined to a resource pool.
- Re-ordering a PIF configured as the host's management interface requires additional steps which are included in the example below. Because the management interface must first be disabled the commands must be entered directly on the host console.

For the example configuration shown above use the following steps to change the NIC ordering so that eth0 corresponds to the device with a MAC address of 00:19:bb:2d:7e:7a:

1. Use XenCenter or the `vm-shutdown` command to shut down all VMs in the pool to force existing VIFs to be unplugged from their networks.

```
xe vm-shutdown uuid=<vm_uuid>
```

2. Use the `host-management-disable` command to disable the management interface:

```
xhost-management-disable
```

3. Use the `pif-forget` command to remove the two incorrect PIF records:

```
xpif-forget uuid=1ef8209d-5db5-cf69-3fe6-0e8d24f8f518
xpif-forget uuid=829fd476-2bbb-67bb-139f-d607c09e9110
```

4. Use the `pif-introduce` command to re-introduce the devices with the desired naming:

```
xpif-introduce device=eth0 host-uuid=<host_uuid> mac=00:19:bb:2d:7e:7a
xpif-introduce device=eth1 host-uuid=<host_uuid> mac=00:19:bb:2d:7e:8a
```

5. Use the `pif-list` command again to verify the new configuration:

```
xpif-list params=uuid,device,MAC
```
6. Use the `pif-reconfigure-ip` command to reset the management interface IP addressing configuration. See Chapter 6, Command line interface for details on the parameters of the `pif-reconfigure-ip` command.

```
xepif-reconfigure-ip uuid=<728d9e7f-62ed-a477-2c71-3974d75972eb> mode= dhcp
```

7. Use the `host-management-reconfigure` command to set the management interface to the desired PIF and re-enable external management connectivity to the host:

```
xehost-management-reconfigure pif-uuid=<728d9e7f-62ed-a477-2c71-3974d75972eb>
```

**Networking Troubleshooting**

If you are having problems with configuring networking, first ensure that you have not directly modified any of the control domain `ifcfg-*` files directly. These files are directly managed by the control domain host agent, and changes will be overwritten.

**Diagnosing network corruption**

Some models of network cards require firmware upgrades from the vendor to work reliably under load, or when certain optimizations are turned on. If you are seeing corrupted traffic to VMs, then you should first try to obtain the latest recommended firmware from your vendor and apply a BIOS update.

If the problem still persists, then you can use the CLI to disable receive/transmit offload optimizations on the physical interface. Be aware that this can result in a performance loss and/or increased CPU usage.

First, determine the UUID of the physical interface. You can filter on the `device` field as follows:

```
xepif-list device=eth0
```

Next, set the following parameter on the PIF to disable TX offload:

```
xepif-param-set uuid=<pif_uuid> other_config:ethtool-tx=off
```

Finally, re-plug the PIF or reboot the host for the change to take effect.

**Recovering from a bad network configuration**

In some cases it is possible to render networking unusable by creating an incorrect configuration. This is particularly true when attempting to make network configuration changes on a member XenServer host.

If a loss of networking occurs, the following notes may be useful in recovering and regaining network connectivity:

- Citrix recommends that you ensure networking configuration is set up correctly before creating a resource pool, as it is usually easier to recover from a bad configuration in a non-pooled state.

- The `host-management-reconfigure` and `host-management-disable` commands affect the XenServer host on which they are run and so are not suitable for use on one host in a pool to change the configuration of another. Run these commands directly on the console of the XenServer host to be affected, or use the `xe -s, -u, and -pw` remote connection options.

- When the `xapi` service starts, it will apply configuration to the management interface first. The name of the management interface is saved in the `/etc/xensource-inventory` file. In extreme cases, you can stop the `xapi` service by running `service xapi stop` at the console, edit the inventory file to set the management interface to a safe default, and then ensure that the `ifcfg` files in `/etc/sysconfig/network-scripts` have correct config-
urations for a minimal network configuration (including one interface and one bridge; for example, eth0 on the xenbr0 bridge).
Chapter 5. Monitoring and managing XenServer

XenServer and XenCenter provide access to alerts that are generated when noteworthy things happen. XenCenter provides various mechanisms of grouping and maintaining metadata about managed VMs, hosts, storage repositories, and so on.

Alerts

XenServer generates alerts for the following events.

Configurable Alerts:

- New XenServer patches available
- New XenServer version available
- New XenCenter version available

Alerts generated by XenCenter:

<table>
<thead>
<tr>
<th>Alert</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XenCenter old</td>
<td>the XenServer expects a newer version but can still connect to the current version</td>
</tr>
<tr>
<td>XenCenter out of date</td>
<td>XenCenter is too old to connect to XenServer</td>
</tr>
<tr>
<td>XenServer out of date</td>
<td>XenServer is an old version that the current XenCenter cannot connect to</td>
</tr>
<tr>
<td>License expired alert</td>
<td>your XenServer license has expired</td>
</tr>
<tr>
<td>Missing IQN alert</td>
<td>XenServer uses iSCSI storage but the host IQN is blank</td>
</tr>
<tr>
<td>Duplicate IQN alert</td>
<td>XenServer uses iSCSI storage, and there are duplicate host IQNs</td>
</tr>
</tbody>
</table>

Alerts generated by XenServer:

- ha_host_failed
- ha_host_was_fenced
- ha_network_bonding_error
- ha_pool_drop_in_plan_exists_for
- ha_pool_overcommitted
- ha_protected_vm_restart_failed
- ha_statefile_lost
- host_clock_skew_detected
- host_sync_data_failed
- license_does_not_support_pooling
- pbd_plug_failed_on_server_start
• pool_master_transition

The following alerts appear on the performance graphs in XenCenter. See the XenCenter online help for more information:
• vm_cloned
• vm_crashed
• vm_rebooted
• vm_resumed
• vm_shutdown
• vm_started
• vm_suspended

**Customizing Alerts**

The performance monitoring `perfmon` runs once every 5 minutes and requests updates from XenServer which are averages over 1 minute, but these defaults can be changed in `/etc/sysconfig/perfmon`.

Every 5 minutes `perfmon` reads updates of performance variables exported by the XAPI instance running on the same host. These variables are separated into one group relating to the host itself, and a group for each VM running on that host. For each VM and also for the host, `perfmon` reads in the `other-config:perfmon` parameter and uses this string to determine which variables it should monitor, and under which circumstances to generate a message.

`vm:other-config:perfmon` and `host:other-config:perfmon` values consist of an XML string like the one below:

```xml
<config>
  <variable>
    <name value="cpu_usage"/>
    <alarm_trigger_level value="LEVEL"/>
  </variable>
  <variable>
    <name value="network_usage"/>
    <alarm_trigger_level value="LEVEL"/>
  </variable>
</config>
```

**Valid VM Elements**

- **name**
  - what to call the variable (no default)
- **alarm_priority**
  - the priority of the messages generated (default 5)
- **alarm_trigger_level**
  - level of value that triggers an alarm (no default)
- **alarm_trigger_sense**
  - high if `alarm_trigger_level` is a max, otherwise low (default high)
- **alarm_trigger_period**
  - number of seconds of 'bad' values before an alarm is sent (default 60)
- **alarm_auto_inhibit_period**
  - number of seconds this alarm disabled after an alarm is sent (default 3600)
- **consolidation_fn**
  - how to combine variables from rrd_updates into one value (default sum - other choice is average)
rrd_regex
matches the names of variables from (xe vm-data-sources-list uuid=\(<\text{vmuuid}\>) used to compute value (only has defaults for cpu_usage, network_usage, and disk_usage).

**Valid Host Elements**

- **name**: what to call the variable (no default)
- **alarm_priority**: the priority of the messages generated (default 5)
- **alarm_trigger_level**: level of value that triggers an alarm (no default)
- **alarm_trigger_sense**: high if \(\text{alarm_trigger_level}\) is a max, otherwise low. (default high)
- **alarm_trigger_period**: number of seconds of 'bad' values before an alarm is sent (default 60)
- **alarm_auto_inhibit_period**: number of seconds this alarm disabled after an alarm is sent (default 3600)
- **consolidation_fn**: how to combine variables from \(\text{rrd_updates}\) into one value (default sum - other choice is average)

rrd_regex
matches the names of variables from \(\text{xe host-data-sources-list uuid=}\(<\text{host_uuid}\>)\) used to compute value (only has defaults for cpu_usage and network_usage).

**Configuring Email Alerts**

Alerts generated from XenServer can also be automatically e-mailed to the resource pool administrator, in addition to being visible from the XenCenter GUI. To configure this, specify the email address and SMTP server:

```
pool:other-config:mail-destination=\(<\text{joe.bloggs@domain.tld}\> 
pool:other-config:ssmtp-mailhub=\(<\text{smtp.domain.tld[:port]}\>
```

You can also specify the minimum value of the priority field in the message before the email will be sent:

```
pool:other-config:mail-min-priority=\(<\text{level}\>
```

The default priority level is 5.

**Note**

Some SMTP servers only forward mails with addresses that use FQDNs. If you find that emails are not being forwarded it may be for this reason, in which case you can set the server hostname to the FQDN so this is used when connecting to your mail server.

**Custom Fields and Tags**

XenCenter supports the creation of tags and custom fields, which allows for organization and quick searching of VMs, storage and so on. See the XenCenter online help for more information.
Custom Searches

XenCenter supports the creation of customized searches. Searches can be exported and imported, and the results of a search can be displayed in the navigation pane. See the XenCenter online help for more information.
Chapter 6. Command line interface

This chapter describes the XenServer command line interface (CLI). The `xe` CLI enables the writing of scripts for automating system administration tasks and allows integration of XenServer into an existing IT infrastructure.

The `xe` command line interface is installed by default on XenServer hosts and is included with XenCenter. A stand-alone remote CLI is also available for Linux.

On Windows, the `xe.exe` CLI executable is installed along with XenCenter. To use it, open a Windows Command Prompt and change directories to the directory where the file resides (typically \Program Files\XenSource\XenCenter), or add its installation location to your system path.

On Linux, you can install the stand-alone `xe` CLI executable from the RPM named `xe-cli-5.0.0-10918p.i386.rpm` on the Linux Pack CD, as follows:

```
rpm -ivh xe-cli-5.0.0-10918p.i386.rpm
```

Basic help is available for CLI commands on-host by typing:

```
xe help command
```

A list of the most commonly-used `xe` commands is displayed if you type:

```
xe help
```

or a list of all `xe` commands is displayed if you type:

```
xe help --all
```

Basic xe syntax

The basic syntax of all XenServer `xe` CLI commands is:

```
xe <command-name> <argument=value> <argument=value> ...
```

Each specific command contains its own set of arguments that are of the form `argument=value`. Some commands have required arguments, and most have some set of optional arguments. Typically a command will assume default values for some of the optional arguments when invoked without them.

If the `xe` command is executed remotely, additional connection and authentication arguments are used. These arguments also take the form `argument=argument_value`.

The `server` argument is used to specify the hostname or IP address. The `username` and `password` arguments are used to specify credentials. A `password-file` argument can be specified instead of the password directly. In this case an attempt is made to read the password from the specified file (stripping CRs and LFs off the end of the file if necessary), and use that to connect. This is more secure than specifying the password directly at the command line.

The optional `port` argument can be used to specify the agent port on the remote XenServer host (defaults to 443).

Example: On the local XenServer host:
xe vm-list

Example: On the remote XenServer host:

```
exe vm-list -user <username> -password <password> -server <hostname>
```

Shorthand syntax is also available for remote connection arguments:

- **-u**  username
- **-pw** password
- **-pf** password file
- **-p**  port
- **-s**  server

Example: On a remote XenServer host:

```
exe vm-list -u <myuser> -pw <mypassword> -s <hostname>
```

Arguments are also taken from the environment variable **XE_EXTRA_ARGS**, in the form of comma-separated key/value pairs. For example, in order to enter commands on one XenServer host that are run on a remote XenServer host, you could do the following:

```
export XE_EXTRA_ARGS="server=jeffbeck, port=443, username=root, password=pass"
```

and thereafter you would not need to specify the remote XenServer host parameters in each xe command you execute.

Using the **XE_EXTRA_ARGS** environment variable also enables tab completion of xe commands when issued against a remote XenServer host, which is disabled by default.

**Special characters and syntax**

To specify argument/value pairs on the **xe** command line, write

```
argument=value
```

without quotes, as long as value doesn't have any spaces in it. There should be no whitespace in between the argument name, the equals sign (=), and the value. Any argument not conforming to this format will be ignored.

For values containing spaces, write:

```
argument="value with spaces"
```

If you use the CLI while logged into a XenServer host, commands have a tab completion feature similar to that in the standard Linux bash shell. If you type, for example

```
exe vm-
```
and then press the **TAB** key, the rest of the command will be displayed when it is unambiguous. If more than one command begins with `vm-l`, hitting **TAB** a second time will list the possibilities. This is particularly useful when specifying object UUIDs in commands.

**Note**

When executing commands on a remote XenServer host, tab completion does not normally work. However if you put the server, username, and password in an environment variable called `XE_EXTRA_ARGS` on the machine from which you are entering the commands, tab completion is enabled. See the section called “Basic xe syntax” for details.

**Command types**

Broadly speaking, the CLI commands can be split in two halves: Low-level commands concerned with listing and parameter manipulation of API objects, and higher level commands for interacting with VMs or hosts in a more abstract level. The low-level commands are:

- `<class>`-list
- `<class>`-param-get
- `<class>`-param-set
- `<class>`-param-list
- `<class>`-param-add
- `<class>`-param-remove
- `<class>`-param-clear

where `<class>` is one of:

- bond
- console
- host
- host-crashdump
- host-cpu
- network
- patch
- pbd
- pif
- pool
- sm
- sr
- task
- template
- vbd
- vdi
- vif
Parameter types

The objects that are addressed with the xe commands have sets of parameters that identify them and define their states.

Most parameters take a single value. For example, the name-label parameter of a VM contains a single string value. In the output from parameter list commands such as `xe vm-param-list`, such parameters have an indication in parentheses that defines whether they can be read and written to, or are read-only. For example, the output of `xe vm-param-list` on a specified VM might have the lines

```
user-version ( RW): 1
is-control-domain ( RO): false
```

The first parameter, `user-version`, is writeable and has the value 1. The second, `is-control-domain`, is read-only and has a value of false.

The two other types of parameters are multi-valued. A set parameter contains a list of values. A map parameter is a set of key/value pairs. As an example, look at the following excerpt of some sample output of the `xe vm-param-list` on a specified VM:

```
platform (MRW): acpi: true; apic: true; pae: true; nx: false
allowed-operations (SRO): pause; clean_shutdown; clean_reboot; \
hard_shutdown; hard_reboot; suspend
```

The `platform` parameter has a list of items that represent key/value pairs. The key names are followed by a colon character (:). Each key/value pair is separated from the next by a semicolon character (;). The M preceding the RW indicates that this is a map parameter and is readable and writeable. The `allowed-operations` parameter has a list that makes up a set of items. The S preceding the RO indicates that this is a set parameter and is readable but not writeable.

In xe commands where you want to filter on a map parameter, or set a map parameter, use the separator : (colon) between the map parameter name and the key/value pair. For example, to set the value of the `foo` key of the `other-config` parameter of a VM to `baa`, the command would be

```
xe vm-param-set uuid=<VM uuid> other-config:foo=baa
```

Note

In previous releases the separator - (dash) was used in specifying map parameters. This syntax still works but is deprecated.

Low-level param commands

There are several commands for operating on parameters of objects: `<class>-param-get`, `<class>-param-set`, `<class>-param-add`, `<class>-param-remove`, `<class>-param-clear`, and `<class>-param-list`. Each of these takes a `uuid` parameter to specify the particular object. Since these are considered low-level commands, they must be addressed by UUID and not by the VM name label.
<class>-param-list uuid=<uuid>

Lists all of the parameters and their associated values. Unlike the class-list command, this will list the values of "expensive" fields.

<class>-param-get uuid=<uuid> param-name=<parameter> [param-key=<key>]

Returns the value of a particular parameter. If the parameter is a map, specifying the param-key will get the value associated with that key in the map. If param-key is not specified, or if the parameter is a set, it will return a string representation of the set or map.

<class>-param-set uuid=<uuid> param=<value>...

Sets the value of one or more parameters.

<class>-param-add uuid=<uuid> param-name=<parameter> [ <key>=<value> ... ] [param-key=<key>]

Adds to either a map or a set parameter. If the parameter is a map, add key/value pairs using the <key>=<value> syntax. If the parameter is a set, add keys with the <param-key>=<key> syntax.

<class>-param-remove uuid=<uuid> param-name=<parameter> param-key=<key>

Removes either a key/value pair from a map, or a key from a set.

<class>-param-clear uuid=<uuid> param-name=<parameter>

Completely clears a set or a map.

Low-level list commands

The <class>-list command lists the objects of type <class>. By default it will list all objects, printing a subset of the parameters. This behavior can be modified in two ways: it can filter the objects so that it only outputs a subset, and the parameters that are printed can be modified.

To change the parameters that are printed, the argument params should be specified as a comma-separated list of the required parameters, e.g.:

```
xe vm-list params=name-label,other-config
```

Alternatively, to list all of the parameters, use the syntax:

```
xe vm-list params=all
```

Note that some parameters that are expensive to calculate will not be shown by the list command. These will be shown as e.g.:

```
allowed-VBD-devices (SRO): <expensive field>
```

In order to obtain these fields, use either the command <class>-param-list or <class>-param-get

To filter the list, the CLI will match parameter values with those specified on the command-line, only printing object that match all of the specified constraints. For example:

```
xe vm-list HVM-boot-policy="BIOS order" power-state=halted
```

will only list those VMs for which both the field power-state has the value halted, and for which the field HVM-boot-policy has the value BIOS order.

It is also possible to filter the list based on the value of keys in maps, or on the existence of values in a set. The syntax for the first of these is map-name:key=value, and the second is set-name:contains=value
For scripting, a useful technique is passing `--minimal` on the command line, causing `xe` to print only the first field in a comma-separated list. For example, the command `xe vm-list --minimal` on a XenServer host with three VMs installed gives the three UUIDs of the VMs, for example:

```
a85d6717-7264-d00e-069b-3b1d19d56ad9,aaa3eec5-9499-bcf3-4c03-af10baea96b7, \
42c044de-df69-4b30-89d9-2c199564581d
```

## xe command reference

This section provides a reference to the `xe` commands. They are grouped by objects that the commands address, and listed alphabetically.

### Bonding commands

Commands for working with network bonds, for resilience with physical interface failover. See the section called “Creating NIC bonds on a standalone host” for details.

The bond object is a reference object which glues together `master` and `member` PIFs. The master PIF is the bonding interface which must be used as the overall PIF to refer to the bond. The member PIFs are a set of 2 or more physical interfaces which have been combined into the high-level bonded interface.

### Bond parameters

Bonds have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>unique identifier/object reference for the bond</td>
<td>read only</td>
</tr>
<tr>
<td>master</td>
<td>UUID for the master bond PIF</td>
<td>read only</td>
</tr>
<tr>
<td>members</td>
<td>set of UUIDs for the underlying bonded PIFs</td>
<td>read only set parameter</td>
</tr>
</tbody>
</table>

**bond-create**

```
bond-create network-uuid=<network_uuid> pif-uuids=<pif_uuid_1,pif_uuid_2,...>
```

Create a bonded network interface on the network specified from a list of existing PIF objects. The command will fail if PIFs are in another bond already, if any member has a VLAN tag set, if the referenced PIFs are not on the same XenServer host, or if fewer than 2 PIFs are supplied.

**bond-destroy**

```
host-bond-destroy uuid=<bond_uuid>
```

Delete a bonded interface specified by its UUID from the XenServer host.

### CD commands

Commands for working with physical CD/DVD drives on XenServer hosts.
## CD parameters

CDs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>unique identifier/object reference for the CD</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>Name for the CD</td>
<td>read/write</td>
</tr>
<tr>
<td>name-description</td>
<td>Description text for the CD</td>
<td>read/write</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>A list of the operations that can be performed on this CD</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>current-operations</td>
<td>A list of the operations that are currently in progress on this CD</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>sr-uuid</td>
<td>The unique identifier/object reference for the SR this CD is part of</td>
<td>read only</td>
</tr>
<tr>
<td>sr-name-label</td>
<td>The name for the SR this CD is part of</td>
<td>read only</td>
</tr>
<tr>
<td>vbd-uuids</td>
<td>A list of the unique identifiers for the VBDs on VMs that connect to this CD</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>crashdump-uuids</td>
<td>Not used on CDs since crashdumps cannot be written to them</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>virtual-size</td>
<td>Size of the CD as it appears to VMs (in bytes)</td>
<td>read only</td>
</tr>
<tr>
<td>physical-utilisation</td>
<td>amount of physical space that the CD image is currently taking up on the SR (in bytes)</td>
<td>read only</td>
</tr>
<tr>
<td>type</td>
<td>Set to User for CDs</td>
<td>read only</td>
</tr>
<tr>
<td>sharable</td>
<td>Whether or not the storage is sharable. Always true for CDs.</td>
<td>read only</td>
</tr>
<tr>
<td>read-only</td>
<td>Whether the CD is read-only, if false, the device is writeable. Always true for CDs.</td>
<td>read only</td>
</tr>
<tr>
<td>storage-lock</td>
<td>true if this disk is locked at the storage level</td>
<td>read only</td>
</tr>
<tr>
<td>parent</td>
<td>Reference to the parent disk, if this CD is part of a chain</td>
<td>read only</td>
</tr>
<tr>
<td>missing</td>
<td>true if SR scan operation reported this CD as not present on disk</td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters for the CD</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>location</td>
<td>The path on which the device is mounted</td>
<td>read only</td>
</tr>
<tr>
<td>managed</td>
<td>true if the device is managed</td>
<td>read only</td>
</tr>
</tbody>
</table>
### cd-list

cd-list [params=<param1,param2,...>] [parameter=<parameter_value>...]  

List the CDs and ISOs (CD image files) on the XenServer host or pool, filtering on the optional argument `params`.

If the optional argument `params` is used, the value of params is a string containing a list of parameters of this object that you want to display. Alternatively, you can use the keyword `all` to show all parameters. If `params` is not used, the returned list shows a default subset of all available parameters.

Optional arguments can be any number of the CD parameters listed at the beginning of this section.

### Console commands

Commands for working with consoles.

The console objects can be listed with the standard object listing command (`xe console-list`), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

### Console parameters

Consoles have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the console</td>
<td>read only</td>
</tr>
<tr>
<td>vm-uuid</td>
<td>The unique identifier/object reference of the VM this console is open on</td>
<td>read only</td>
</tr>
<tr>
<td>vm-name-label</td>
<td>The name of the VM this console is open on</td>
<td>read only</td>
</tr>
<tr>
<td>protocol</td>
<td>Protocol this console uses. Possible values are <code>vt100</code>: VT100 terminal, <code>rfb</code>: Remote FrameBuffer protocol (as used in VNC), or <code>rdp</code>: Remote Desktop Protocol</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>location</td>
<td>URI for the console service</td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters for the console.</td>
<td>read/write map parameter</td>
</tr>
</tbody>
</table>

**Event commands**

Commands for working with events.

**Event classes**

Event classes are listed in the following table:

<table>
<thead>
<tr>
<th>Class name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pool</td>
<td>A pool of physical hosts</td>
</tr>
<tr>
<td>vm</td>
<td>A Virtual Machine</td>
</tr>
<tr>
<td>host</td>
<td>A physical host</td>
</tr>
<tr>
<td>network</td>
<td>A virtual network</td>
</tr>
<tr>
<td>vif</td>
<td>A virtual network interface</td>
</tr>
<tr>
<td>pif</td>
<td>A physical network interface</td>
</tr>
<tr>
<td>sr</td>
<td>A storage repository</td>
</tr>
<tr>
<td>vdi</td>
<td>A virtual disk image</td>
</tr>
<tr>
<td>vbd</td>
<td>A virtual block device</td>
</tr>
<tr>
<td>pbd</td>
<td>The physical block devices through which hosts access SRs</td>
</tr>
</tbody>
</table>

**event-wait**

event-wait class=<class_name> [<param-name>=<param_value>] [<param-name>=/=param_value>]

Block other commands from executing until an object exists that satisfies the conditions given on the command line. 
x=y means "wait for field x to take value y", and x=/=y means "wait for field x to take any value other than y".

*Example:* wait for a specific VM to be running

```
xe event-wait class=vm name-label=myvm power-state=running
```

blocks until a VM called *myvm* is in the *power-state"*"running."

*Example:* wait for a specific VM to reboot:

```
xe event-wait class=vm uuid=$VM start-time=/=$(xe vm-list uuid=$VM params=start-time --minimal)
```

blocks until a VM with UUID $VM reboots (i.e. has a different *start-time* value).
The class name can be any of the Event classes listed at the beginning of this section, and the parameters can be any of those listed in the CLI command class-param-list.

**Host (XenServer host) commands**

Commands for interacting with XenServer host.

XenServer hosts are the physical servers running XenServer software. They have VMs running on them under the control of a special privileged Virtual Machine, known as the control domain or domain 0.

The XenServer host objects can be listed with the standard object listing command (xe host-list, xe host-cpu-list, and xe host-crashdump-list), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**Host selectors**

Several of the commands listed here have a common mechanism for selecting one or more XenServer hosts on which to perform the operation. The simplest is by supplying the argument host=<uuid_or_name_label>. XenServer hosts can also be specified by filtering the full list of hosts on the values of fields. For example, specifying enabled=true will select all XenServer hosts whose enabled field is equal to true. Where multiple XenServer hosts are matching, and the operation can be performed on multiple XenServer hosts, the option --multiple must be specified to perform the operation. The full list of parameters that can be matched is described at the beginning of this section, and can be obtained by the command xe host-list params=all. If no parameters to select XenServer hosts are given, the operation will be performed on all XenServer hosts.

**Host parameters**

XenServer hosts have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the XenServer host</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the XenServer host</td>
<td>read/write</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the XenServer host</td>
<td>read only</td>
</tr>
<tr>
<td>enabled</td>
<td>false if disabled which prevents any new VMs from starting on them, which prepares the XenServer hosts to be shut down or re-booted; true if the host is currently enabled</td>
<td>read only</td>
</tr>
<tr>
<td>API-version-major</td>
<td>major version number</td>
<td>read only</td>
</tr>
<tr>
<td>API-version-minor</td>
<td>minor version number</td>
<td>read only</td>
</tr>
<tr>
<td>API-version-vendor</td>
<td></td>
<td>read only</td>
</tr>
<tr>
<td>API-version-vendor-im-plementation</td>
<td>details of vendor implementation</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>logging</td>
<td>logging configuration</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>suspend-image-sr-uuid</td>
<td>the unique identifier/object reference for the SR where suspended images are put</td>
<td>read/write</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>crash-dump-sr-uuid</td>
<td>the unique identifier/object reference for the SR where crash dumps are put</td>
<td>read/write</td>
</tr>
<tr>
<td>software-version</td>
<td>list of versioning parameters and their values</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>capabilities</td>
<td>list of Xen versions that the XenServer host can run</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters for the XenServer host</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>hostname</td>
<td>XenServer host hostname</td>
<td>read only</td>
</tr>
<tr>
<td>address</td>
<td>XenServer host IP address</td>
<td>read only</td>
</tr>
<tr>
<td>supported-bootloaders</td>
<td>list of bootloaders that the XenServer host supports, for example, pygrub, eliloader</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>memory-total</td>
<td>total amount of physical RAM on the XenServer host, in bytes</td>
<td>read only</td>
</tr>
<tr>
<td>memory-free</td>
<td>total amount of physical RAM remaining that can be allocated to VMs, in bytes</td>
<td>read only</td>
</tr>
<tr>
<td>host-metrics-live</td>
<td>true if the host is operational</td>
<td>read only</td>
</tr>
<tr>
<td>logging</td>
<td>The <code>syslog_destination</code> key can be set to the hostname of a remote listening syslog service.</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>list of the operations allowed in this state. This list is advisory only and the server state may have changed by the time this field is read by a client.</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>current-operations</td>
<td>links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>patches</td>
<td>Set of host patches</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>blobs</td>
<td>Binary data store</td>
<td>read only</td>
</tr>
<tr>
<td>memory-free-computed</td>
<td>A conservative estimate of the maximum amount of memory free on a host</td>
<td>read only</td>
</tr>
<tr>
<td>ha-statefiles</td>
<td>The UUID(s) of all HA statefiles</td>
<td>read only</td>
</tr>
<tr>
<td>ha-network-peers</td>
<td>The UUIDs of all hosts that could host the VMs on this host in case of failure</td>
<td>read only</td>
</tr>
</tbody>
</table>

XenServer hosts contain some other objects that also have parameter lists.
CPUs on XenServer hosts have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the CPU</td>
<td>read only</td>
</tr>
<tr>
<td>number</td>
<td>the number of the physical CPU core within the XenServer host</td>
<td>read only</td>
</tr>
<tr>
<td>vendor</td>
<td>the vendor string for the CPU name, for example, &quot;GenuineIntel&quot;</td>
<td>read only</td>
</tr>
<tr>
<td>speed</td>
<td>The CPU clock speed, in Hz</td>
<td>read only</td>
</tr>
<tr>
<td>modelname</td>
<td>the vendor string for the CPU model, for example, &quot;Intel(R) Xeon(TM) CPU 3.00GHz&quot;</td>
<td>read only</td>
</tr>
<tr>
<td>stepping</td>
<td>the CPU revision number</td>
<td>read only</td>
</tr>
<tr>
<td>flags</td>
<td>the flags of the physical CPU (a decoded version of the features field)</td>
<td>read only</td>
</tr>
<tr>
<td>utilisation</td>
<td>the current CPU utilization</td>
<td>read only</td>
</tr>
<tr>
<td>host-uuid</td>
<td>the UUID if the host the CPU is in</td>
<td>read only</td>
</tr>
<tr>
<td>model</td>
<td>the model number of the physical CPU</td>
<td>read only</td>
</tr>
<tr>
<td>family</td>
<td>the family (number) of the physical CPU</td>
<td>read only</td>
</tr>
</tbody>
</table>

Crash dumps on XenServer hosts have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the crashdump</td>
<td>read only</td>
</tr>
<tr>
<td>host</td>
<td>XenServer host the crashdump corresponds to</td>
<td>read only</td>
</tr>
<tr>
<td>timestamp</td>
<td>Timestamp of the date and time that the crashdump occurred, in the form yyyy-mm-dd-hhmmss-ABC, where ABC is the time-zone indicator, for example, GMT</td>
<td>read only</td>
</tr>
<tr>
<td>size</td>
<td>size of the crashdump, in bytes</td>
<td>read only</td>
</tr>
</tbody>
</table>

**host-backup**

```
host-backup file-name=<backup_filename> [ <host-selector>=<host_selector_value> ... ]
```

Download a backup of the control domain of the specified XenServer host to the machine that the command is invoked from, and saving it there as a file with the name `file-name`

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see `host selectors` above). Optional arguments can be any number of the `host selectors` listed at the beginning of this section.
Caution

While the `xe host-backup` command will work if executed on the local host (that is, without a specific hostname specified), do not use it this way. Doing so would fill up the control domain partition with the backup file, which would be a bad thing. The command should only be used from a remote off-host machine where you have space to hold the backup file.

**host-bugreport-upload**

`host-bugreport-upload [host-selector]=<host_selector_value>... [url]=<destination_url> ] [http-proxy=<http_proxy_name>]`

Generate a fresh bug report (via xen-bugtool, with all optional files included) and upload to Citrix Support ftp site or other location.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

Optional parameters are `http-proxy`: use specified http proxy, and `url`: upload to this destination url. If optional parameters are not used, no proxy server is identified and the destination will be the default Citrix Support ftp site.

**host-crashdump-destroy**

`host-crashdump-destroy uuid=<crashdump_uuid>`

Delete a host crashdump specified by its UUID from the XenServer host.

**host-crashdump-upload**

`host-crashdump-upload uuid=<crashdump_uuid> [url]=<destination_url> ] [http-proxy=<http_proxy_name>]`

Upload a crashdump to the Citrix Support ftp site or other location. If optional parameters are not used, no proxy server is identified and the destination will be default Citrix Support ftp site. Optional parameters are `http-proxy`: use specified http proxy, and `url`: upload to this destination url.

**host-disable**

`host-disable [host-selector]=<host_selector_value>...]`

Disables specified XenServer hosts, which prevents any new VMs from starting on them. This prepares the XenServer hosts to be shut down or rebooted.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

**host-dmesg**

`host-dmesg [host-selector]=<host_selector_value>...]`

Get a Xen dmesg (the output of the kernel ring buffer) from specified XenServer hosts.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.
**host-emergency-management-reconfigure**

host-emergency-management-reconfigure interface=\<uuid_of_management_interface_pif>\

Reconfigure the management interface of this XenServer host. Use this command only if the XenServer host is in emergency mode, meaning it is a member in a resource pool whose master has disappeared from the network and could not be contacted for some number of retries.

**host-enable**

host-enable [\<host-selector>=\<host_selector_value>...]

Enables the specified XenServer hosts, which allows new VMs to be started on them.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

**host-evacuate**

host-evacuate [\<host-selector>=\<host_selector_value>...]

Disables the selected host, and live migrates all running VMs to other suitable hosts on a pool.

If the evacuated host is the pool master, then another host must be selected to be the pool master. To change the pool master with HA disabled, you need to use the pool-designate-new-master command. See the section called “pool-designate-new-master” for details. With HA enabled, your only option is to shut down the server, which will cause HA to elect a new master at random. See the section called “host-shutdown”.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

**host-forget**

host-forget uuid=\<XenServer host UUID>

The xapi agent forgets about the specified XenServer host without contacting it explicitly.

**Tip**

This command is useful if the XenServer host to "forget" is dead; however, if the XenServer host is live and part of the pool, you should use xe pool-eject instead.

**host-get-system-status**

host-get-system-status filename=\<name_for_status_file>\
[entries=\<comma_separated_list>] [output=\<tar.bz2 | zip>] [\<host-selector>=\<host_selector_value>...]

Download system status information into the specified file. The optional parameter entries is a comma-separated list of system status entries, taken from the capabilities XML fragment returned by the host-get-system-status-capabilities command. See the section called “host-get-system-status-capabilities” for details. If not specified, all system status information is saved in the file. The parameter output may be tar.bz2 (the default) or zip; if this parameter is not specified, the file is saved in tar.bz2 form.
The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above).

**host-get-system-status-capabilities**

host-get-system-status-capabilities [host-selector=<host_selector_value>...]

Get system status capabilities for the specified host(s). The capabilities are returned as an XML fragment that looks something like this:

```xml
<?xml version="1.0" ?> <system-status-capabilities>
    <capability content-type="text/plain" default-checked="yes" key="xenserver-logs" \
        max-size="150425200" max-time="-1" min-size="150425200" min-time="-1" \
        pii="maybe"/>
    <capability content-type="text/plain" default-checked="yes" \
        key="xenserver-install" max-size="51200" max-time="-1" min-size="10240" \
        min-time="-1" pii="maybe"/>
...
</system-status-capabilities>
```

Each capability entity has a number of attributes.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>A unique identifier for the capability.</td>
</tr>
<tr>
<td>content-type</td>
<td>Can be either text/plain or application/data. Indicates whether a UI can render the entries for human consumption.</td>
</tr>
<tr>
<td>default-checked</td>
<td>Can be either yes or no. Indicates whether a UI should select this entry by default.</td>
</tr>
<tr>
<td>min-size, max-size</td>
<td>Indicates an approximate range for the size, in bytes, of this entry. -1 indicates that the size is unimportant.</td>
</tr>
<tr>
<td>min-time, max-time</td>
<td>Indicate an approximate range for the time, in seconds, taken to collect this entry. -1 indicates the time is unimportant.</td>
</tr>
<tr>
<td>pii</td>
<td>Personally identifiable information. Indicates whether the entry would have information that would identify the system owner, or details of their network topology. This is one of:</td>
</tr>
<tr>
<td></td>
<td>• no: no PII will be in these entries</td>
</tr>
<tr>
<td></td>
<td>• yes: PII will likely or certainly be in these entries</td>
</tr>
<tr>
<td></td>
<td>• maybe: you might wish to audit these entries for PII</td>
</tr>
<tr>
<td></td>
<td>• if_customized if the files are unmodified, then they will contain no PII, but since we encourage editing of these files, PII may have been introduced by such customization. This is used in particular for the networking scripts in the control domain.</td>
</tr>
</tbody>
</table>

Passwords are never to be included in any bug report, regardless of any PII declaration.
The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above).

**host-is-in-emergency-mode**

Determine if the host the CLI is talking to is currently in emergency mode. If it is, then `true` will be output, otherwise it will be `false`. This CLI command works directly on slave hosts without a master host needing to be present.

**host-license-add**

```
host-license-add license-file= <path/license_filename>  [host-uuid= <XenServer host UUID>] 
```

Parses a local license file and adds it to the specified XenServer host.

For details on licensing a host, see the chapter "XenServer Licensing" in the *XenServer Installation Guide*.

**host-license-view**

```
host-license-view [host-uuid= <XenServer host UUID>] 
```

Displays the contents of the XenServer host license.

**host-logs-download**

```
host-logs-download [file-name= <logfile_name>] [host-selector= <host_selector_value>...] 
```

Download a copy of the logs of the specified XenServer hosts. The copy is saved by default in a timestamped file named `hostname-yyyy-mm-dd T hh:mm:ssZ.tar.gz`. You can specify a different filename using the optional parameter `file-name`.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

**Caution**

While the `xe host-logs-download` command will work if executed on the local host (that is, without a specific hostname specified), do *not* use it this way. Doing so will clutter the control domain partition with the copy of the logs, which would be a bad thing. The command should only be used from a remote off-host machine where you have space to hold the copy of the logs.

**host-management-disable**

```
host-management-disable 
```

Disables the host agent listening on an external management network interface and disconnects all connected API clients (such as the XenCenter). Operates directly on the XenServer host the CLI is connected to, and is not forwarded to the pool master if applied to a member XenServer host.
Warning

Be extremely careful when using this CLI command off-host, since once it is run it will not be possible to connect to the control domain remotely over the network to re-enable it.

**host-management-reconfigure**

```plaintext
host-management-reconfigure [interface=<device> ] | [pif-uuid=<uuid> ]
```

Reconfigures the XenServer host to use the specified network interface as its management interface, which is the interface that is used to connect to the XenCenter. The command rewrites the MANAGEMENT_INTERFACE key in `/etc/xensource-inventory`.

If the device name of an interface (which must have an IP address) is specified, the XenServer host will immediately rebind. This works both in normal and emergency mode.

If the UUID of a PIF object is specified, the XenServer host will determine which IP address to rebind to itself. It must not be in emergency mode when this command is executed.

Warning

Be careful when using this CLI command off-host and ensure you have network connectivity on the new interface (by using `xe pif-reconfigure` to set one up first). Otherwise, subsequent CLI commands will not be able to reach the XenServer host.

**host-reboot**

```plaintext
host-reboot [ <host-selector>=<host_selector_value> ... ]
```

Reboot the specified XenServer hosts. The specified XenServer hosts must be disabled first using the `xe host-disable` command, otherwise a "HOST_IN_USE" error message is displayed.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see [host selectors](#)). Optional arguments can be any number of the [host selectors](#) listed at the beginning of this section.

If the specified XenServer hosts are members of a pool, the loss of connectivity on shutdown will be handled and the pool will recover when the XenServer hosts returns. If you shut down a pool member, other members and the master will continue to function. If you shut down the master, the pool will be out of action until the master is rebooted and back on line, at which point the members will reconnect and synchronize with the master, or until you make one of the members into the master.

**host-restore**

```plaintext
host-restore [file-name=<backup_filename>] [ <host-selector>=<host_selector_value> ... ]
```

Restore a backup named `file-name` of the XenServer host control software. Note that the use of the word "restore" here does not mean a full restore in the usual sense, it merely means that the compressed backup file has been uncompressed and unpacked onto the secondary partition. After you've done a `xe host-restore`, you have to boot the Install CD and use its "restore secondary partition to primary partition" option.
The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

**host-set-hostname-live**

`host-set-hostname host-uuid=<uuid_of_host> hostname=<new_hostname>`

Change the hostname of the XenServer host specified by `host-uuid`. This command persistently sets both the hostname in the control domain's agent database and the actual Linux hostname of the XenServer host. Note that `hostname` is not the same as the value of the `name_label` field.

**host-shutdown**

`host-shutdown [host-selector]=<host_selector_value>...`

Shut down the specified XenServer hosts. The specified XenServer hosts must be disabled first using the `xe host-disable` command, otherwise a "HOST_IN_USE" error message is displayed.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

If the specified XenServer hosts are members of a pool, the loss of connectivity on shutdown will be handled and the pool will recover when the XenServer hosts returns. If you shut down a pool member, other members and the master will continue to function. If you shut down the master, the pool will be out of action until the master is rebooted and back on line, at which point the members will reconnect and synchronize with the master, or until one of the members is made into the master. If HA is enabled for the pool, one of the members will be made into a master automatically. If HA is disabled, you must manually make the desired server into the master with the `pool-designate-new-master` command. See the section called "pool-designate-new-master".

**host-syslog-reconfigure**

`host-syslog-reconfigure [host-selector]=<host_selector_value>...`

Reconfigure the syslog daemon on the specified XenServer hosts. This command will apply the configuration information defined in the host `logging` parameter.

The host(s) on which this operation should be performed are selected via the standard selection mechanism (see host selectors above). Optional arguments can be any number of the host selectors listed at the beginning of this section.

**Log commands**

Commands for working with logs.

**log-get-keys**

`log-get-keys`

List the keys of all of the logging subsystems.

**log-reopen**

`log-reopen`
Reopen all loggers. Use this for rotating log files.

**log-set-output**

```
log-set-output output=nil | stderr | file:<filename> | syslog:<sysloglocation> [key=<key>] [level= debug | info | warning | error]
```

Set the output of the specified logger. Log messages are filtered by the subsystem in which they originated and the log level of the message. For example, debug logging messages from the storage manager can be sent to a file via the following command:

```
xe log-set-output key=sm level=debug output=file:/tmp/sm.log
```

The optional parameter `key` specifies the particular logging subsystem. If this parameter is not set, it will default to all logging subsystems.

The optional parameter `level` specifies the logging level. Valid values are:

- debug
- info
- warning
- error

**Network commands**

Commands for working with networks.

The network objects can be listed with the standard object listing command (`xe network-list`), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**Network parameters**

Networks have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the network</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the network</td>
<td>read write</td>
</tr>
<tr>
<td>name-description</td>
<td>The description text of the network</td>
<td>read write</td>
</tr>
<tr>
<td>VIF-uuids</td>
<td>A list of unique identifiers of the VIFs (virtual network interfaces) that are attached from VMs to this network</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>PIF-uuids</td>
<td>A list of unique identifiers of the PIFs (physical network interfaces) that are attached from XenServer hosts to this network</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>bridge</td>
<td>name of the bridge corresponding to this network on the local XenServer host</td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters for the network.</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>blobs</td>
<td>Binary data store</td>
<td>read only</td>
</tr>
</tbody>
</table>

**network-create**

`network-create name-label=<name_for_network> [name-description=<descriptive_text>]`

Creates a new network.

**network-destroy**

`network-destroy uuid=<network_uuid>`

Destroys an existing network.

**Patch (update) commands**

Commands for working with XenServer host patches (updates). These are for the standard non-OEM editions of XenServer for commands relating to updating the OEM edition of XenServer, see the section called “Update commands” for details.

The patch objects can be listed with the standard object listing command (`xe patch-list`), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**Patch parameters**

Patches have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the patch</td>
<td>read only</td>
</tr>
<tr>
<td>host-uuid</td>
<td>The unique identifier for the XenServer host to query</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the patch</td>
<td>read only</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the patch</td>
<td>read only</td>
</tr>
<tr>
<td>applied</td>
<td>Whether or not the patch has been applied; true or false</td>
<td>read only</td>
</tr>
<tr>
<td>size</td>
<td>Whether or not the patch has been applied; true or false</td>
<td>read only</td>
</tr>
</tbody>
</table>
patch-apply
patch-apply uuid=<patch_file_uuid>
Apply the specified patch file.

patch-clean
patch-clean uuid=<patch_file_uuid>
Delete the specified patch file from the XenServer host.

patch-pool-apply
patch-pool-apply uuid=<patch_uuid>
Apply the specified patch to all XenServer hosts in the pool.

patch-precheck
patch-precheck uuid=<patch_uuid> host-uuid=<host_uuid>
Run the prechecks contained within the specified patch on the specified XenServer host.

patch-upload
patch-upload file-name=<patch_filename>
Upload a specified patch file to the XenServer host. This prepares a patch to be applied. On success, the UUID of the uploaded patch is printed out. If the patch has previously been uploaded, a PATCH_ALREADY_EXISTS error is returned instead and the patch is not uploaded again.

PBD commands
Commands for working with PBDs (Physical Block Devices). These are the software objects through which the XenServer host accesses storage repositories (SRs).

The PBD objects can be listed with the standard object listing command (**xe pbd-list**), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

PBD parameters
PBDs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the PBD.</td>
<td>read only</td>
</tr>
<tr>
<td>sr-uuid</td>
<td>the storage repository that the PBD points to</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>device-config</td>
<td>additional configuration information that is provided to the XenServer host's SR-backend-driver</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>currently-attached</td>
<td>Is the SR currently attached on this host? True or False</td>
<td>read only</td>
</tr>
<tr>
<td>host-uuid</td>
<td>UUID of the physical machine on which the PBD is available</td>
<td>read only</td>
</tr>
<tr>
<td>host</td>
<td>The host field is deprecated. Use host-uuid instead.</td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>Additional configuration information.</td>
<td>read/write map parameter</td>
</tr>
</tbody>
</table>

**pbd-create**

```
pbd-create host-uuid=<uuid_of_host> sr-uuid=<uuid_of_sr> [device-config:key=<corresponding_value>...]
```

Create a new PBD on a XenServer host. The read-only `device-config` parameter can only be set on creation as in the following example:

```
To add a mapping of 'path' -> '/tmp', the command line should contain the argument `device-config:path=/tmp`.
```

For a full list of supported device-config key/value pairs on each SR type see the section called “Storage”.

**pbd-destroy**

```
pbd-destroy uuid=<uuid_of_pbd>
```

Destroy the specified PBD.

**pbd-plug**

```
pbd-plug uuid=<uuid_of_pbd>
```

Attempts to plug in the PBD to the XenServer host. If this succeeds, the referenced SR (and the VDIs contained within) should then become visible to the XenServer host.

**pbd-unplug**

```
pbd-unplug uuid=<uuid_of_pbd>
```

Attempts to unplug the PBD from the XenServer host.

**PIF commands**

Commands for working with PIFs (objects representing the physical network interfaces).
The PIF objects can be listed with the standard object listing command (**xe pif-list**), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**PIF parameters**

PIFs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the PIF</td>
<td>read only</td>
</tr>
<tr>
<td>device</td>
<td>machine-readable name of the interface (for example, eth0)</td>
<td>read only</td>
</tr>
<tr>
<td>MAC</td>
<td>the MAC address of the PIF</td>
<td>read only</td>
</tr>
<tr>
<td>physical</td>
<td>If true, the PIF points to an actual physical network interface</td>
<td>read only</td>
</tr>
<tr>
<td>currently-attached</td>
<td>Is the PIF currently attached on this host? true or false</td>
<td>read only</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit of the PIF in bytes.</td>
<td>read only</td>
</tr>
<tr>
<td>VLAN</td>
<td>VLAN tag for all traffic passing through this interface; -1 indicates no VLAN tag is assigned</td>
<td></td>
</tr>
<tr>
<td>bond-master-of</td>
<td>The UUID of the bond this PIF is the master of (if any)</td>
<td>read only</td>
</tr>
<tr>
<td>bond-slave-of</td>
<td>The UUID of the bond this PIF is the slave of (if any)</td>
<td>read only</td>
</tr>
<tr>
<td>management</td>
<td>Is this PIF designated to be a management interface for the control domain</td>
<td>read only</td>
</tr>
<tr>
<td>network-uuid</td>
<td>the unique identifier/object reference of the virtual network to which this PIF is connected</td>
<td></td>
</tr>
<tr>
<td>network-name-label</td>
<td>the name of the of the virtual network to which this PIF is connected</td>
<td>read only</td>
</tr>
<tr>
<td>host-uuid</td>
<td>the unique identifier/object reference of the XenServer host to which this PIF is connected</td>
<td></td>
</tr>
<tr>
<td>host-name-label</td>
<td>the name of the XenServer host to which this PIF is connected</td>
<td>read only</td>
</tr>
<tr>
<td>IP-configuration-mode</td>
<td>Type of network address configuration used; DHCP or static</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>IP</td>
<td>IP address of the PIF, defined here if IP-configuration-mode is static; undefined if DHCP</td>
<td>read only</td>
</tr>
<tr>
<td>netmask</td>
<td>Netmask of the PIF, defined here if IP-configuration-mode is static; undefined if supplied by DHCP</td>
<td>read only</td>
</tr>
<tr>
<td>gateway</td>
<td>Gateway address of the PIF, defined here if IP-configuration-mode is static; undefined if supplied by DHCP</td>
<td>read only</td>
</tr>
<tr>
<td>DNS</td>
<td>DNS address of the PIF, defined here if IP-configuration-mode is static; undefined if supplied by DHCP</td>
<td>read only</td>
</tr>
<tr>
<td>io_read_kbs</td>
<td>average read rate in kB/s for the device</td>
<td>read only</td>
</tr>
<tr>
<td>io_write_kbs</td>
<td>average write rate in kB/s for the device</td>
<td>read only</td>
</tr>
<tr>
<td>carrier</td>
<td>link state for this device</td>
<td>read only</td>
</tr>
<tr>
<td>vendor-id</td>
<td>the ID assigned to NIC's vendor</td>
<td>read only</td>
</tr>
<tr>
<td>vendor-name</td>
<td>the NIC vendor's name</td>
<td>read only</td>
</tr>
<tr>
<td>device-id</td>
<td>the ID assigned by the vendor to this NIC</td>
<td>read only</td>
</tr>
<tr>
<td>device-name</td>
<td>the name assigned by the vendor to this NIC model</td>
<td>read only</td>
</tr>
<tr>
<td>speed</td>
<td>Data transfer rate of the NIC</td>
<td>read only</td>
</tr>
<tr>
<td>duplex</td>
<td>Duplexing mode of the NIC; full or half</td>
<td>read only</td>
</tr>
<tr>
<td>pci-bus-path</td>
<td></td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>additional configuration</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>disallow-unplug</td>
<td>True if this PIF is a dedicated storage NIC, false otherwise</td>
<td>read/write</td>
</tr>
</tbody>
</table>

**Note**

Changes made to the `other-config` fields of a PIF will only take effect after a reboot. Alternately, use the `xe pif-unplug` and `xe pif-plug` commands to cause the PIF configuration to be rewritten.

**pif-forget**

`pif-forget uuid=<uuid_of_pif>`
Destroy the specified PIF object on a particular host.

**pif-introduce**

```
pif-introduce host-uuid=UUID of XenServer host mac=mac_address_for_pif device=machine-readable name of the interface (for example, eth0)
```

Create a new PIF object representing a physical interface on the specified XenServer host.

**pif-plug**

```
pif-plug uuid=uuid_of_pif
```

Attempt to bring up the specified physical interface.

**pif-reconfigure-ip**

```
pif-reconfigure-ip uuid=uuid_of_pif [ mode=dhcp | mode=static ]
gateway=network_gateway_address IP=static_ip_for_this_pif
netmask=netmask_for_this_pif [DNS=dns_address]
```

Modify the IP address of the PIF. For static IP configuration, set the mode parameter to static, with the gateway, IP, and netmask parameters set to the appropriate values. To use DHCP, just set the mode parameter to DHCP and leave the static parameters undefined.

**pif-scan**

```
pif-scan host-uuid=UUID of XenServer host
```

Scan for new physical interfaces on a XenServer host.

**pif-unplug**

```
pif-unplug uuid=uuid_of_pif
```

Attempt to bring down the specified physical interface.

**Pool commands**

Commands for working with pools. A pool is an aggregate of one or more XenServer hosts. A pool uses one or more shared storage repositories so that the VMs running on one XenServer host in the pool can be migrated in near-real time (while still running, without needing to be shut down and brought back up) to another XenServer host in the pool. Each XenServer host is really a pool consisting of a single member by default. When a XenServer host is joined to a pool, it is designated as a member, and the pool it has joined becomes the master for the pool.

The singleton pool object can be listed with the standard object listing command (**xe pool-list**), and its parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**Pool parameters**

Pools have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>the unique identifier/object reference for the pool</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the pool</td>
<td>read/write</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the pool</td>
<td>read/write</td>
</tr>
<tr>
<td>master</td>
<td>the unique identifier/object reference of XenServer host designated as the</td>
<td>read only</td>
</tr>
<tr>
<td></td>
<td>pool's master</td>
<td></td>
</tr>
<tr>
<td>default-SR</td>
<td>the unique identifier/object reference of the default SR for the pool</td>
<td>read/write</td>
</tr>
<tr>
<td>crash-dump-SR</td>
<td>the unique identifier/object reference of the SR where any crash dumps for</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>pool members are saved</td>
<td></td>
</tr>
<tr>
<td>suspend-image-SR</td>
<td>the unique identifier/object reference of the SR where suspended VMs on</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>pool members are saved</td>
<td></td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>for the pool</td>
<td>map parameter</td>
</tr>
<tr>
<td>supported-sr-types</td>
<td>SR types that can be used by this pool</td>
<td>read only</td>
</tr>
<tr>
<td>ha-enabled</td>
<td>True if HA is enabled for the pool, false otherwise</td>
<td>read only</td>
</tr>
<tr>
<td>ha-configuration</td>
<td>Reserved for future use.</td>
<td>read only</td>
</tr>
<tr>
<td>ha-statefiles</td>
<td>Lists the UUIDs of the VDI being used by HA to determine storage health</td>
<td>read only</td>
</tr>
<tr>
<td>ha-host-failures-to-</td>
<td>The number of host failures to tolerate before sending a system alert</td>
<td>read/write</td>
</tr>
<tr>
<td>tolerate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ha-plan-exists-for</td>
<td>The number of hosts failures that can actually be handled, according to the</td>
<td>read only</td>
</tr>
<tr>
<td></td>
<td>calculations of the HA algorithm</td>
<td></td>
</tr>
<tr>
<td>ha-allow-overcommit</td>
<td>True if the pool is allowed to be overcommitted, false otherwise</td>
<td>read/write</td>
</tr>
<tr>
<td>ha-overcommitted</td>
<td>True if the pool is currently overcommitted</td>
<td>read only</td>
</tr>
<tr>
<td>blobs</td>
<td>Binary data store</td>
<td>read only</td>
</tr>
</tbody>
</table>

**pool-designate-new-master**

`pool-designate-new-master host-uuid=UUID of member XenServer host to become new master`

Instruct the specified member XenServer host to become the master of an existing pool. This performs an orderly handover of the role of master host to another host in the resource pool. This command only works when the current master is online, and is not a replacement for the emergency mode commands listed below.
**pool-dump-database**

`pool-dump-database file-name=filename to dump database into (on client)`

Download a copy of the entire pool database and dump it into a file on the client.

**pool-eject**

`pool-eject host-uuid=UUID of XenServer host to eject`

Instruct the specified XenServer host to leave an existing pool.

**pool-emergency-reset-master**

`pool-emergency-reset-master master-address=address of the pool's master XenServer host`

Instruct a slave member XenServer host to reset its master address to the new value and attempt to connect to it. This command should not be run on master hosts.

**pool-emergency-transition-to-master**

Instruct a member XenServer host to become the pool master. This command is only accepted by the XenServer host if it has transitioned to emergency mode, meaning it is a member of a pool whose master has disappeared from the network and could not be contacted for some number of retries.

Note that this command may cause the password of the host to reset if it has been modified since joining the pool (see the section called “User commands”).

**pool-ha-enable**

`pool-ha-enable heartbeat-sr-uuids=SR UUID of the Heartbeat SR`

Enable High-Availability on the resource pool, using the specified SR UUID as the central storage heartbeat repository.

**pool-ha-disable**

`pool-ha-enable`

Disables the High-Availability functionality on the resource pool.

**pool-join**

`pool-join master-address=address master-username=username master-password=password`

Instruct a XenServer host to join an existing pool.

**pool-recover-slaves**

`pool-recover-slaves`
Instruct the pool master to try and reset the master address of all members currently running in emergency mode. This is typically used after `pool-emergency-transition-to-master` has been used to set one of the members as the new master.

**pool-restore-database**

```bash
pool-restore-database file-name=<filename_to_restore_from_(on_client)>
```

Upload a database backup (created with "pool-dump-database") to a pool. On receiving the upload, the master will restart itself with the new database.

**pool-sync-database**

```bash
pool-sync-database
```

Force the pool database to be synchronized across all hosts in the resource pool. This is not necessary in normal operation since the database is regularly replicated automatically, but can be useful for ensuring changes are rapidly replicated after performing a significant set of CLI operations.

## Storage Manager commands

Commands for controlling Storage Manager plugins.

The storage manager objects can be listed with the standard object listing command (`xe sm-list`), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

### SM parameters

SMs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the SM plugin</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the SM plugin</td>
<td>read only</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the SM plugin</td>
<td>read only</td>
</tr>
<tr>
<td>type</td>
<td>The SR type that this plugin connects to</td>
<td>read only</td>
</tr>
<tr>
<td>vendor</td>
<td>Name of the vendor who created this plugin</td>
<td>read only</td>
</tr>
<tr>
<td>copyright</td>
<td>copyright statement for this SM plugin</td>
<td>read only</td>
</tr>
<tr>
<td>required-api-version</td>
<td>Minimum SM API version required on the XenServer host</td>
<td>read only</td>
</tr>
<tr>
<td>configuration</td>
<td>names and descriptions of device configuration keys</td>
<td>read only</td>
</tr>
<tr>
<td>capabilities</td>
<td>capabilities of the SM plugin</td>
<td>read only</td>
</tr>
</tbody>
</table>
Parameter Name | Description | Type
---|---|---
driver-filename | The filename of the SR driver. | read only

**SR commands**

Commands for controlling SRs (storage repositories).

The SR objects can be listed with the standard object listing command (*xe sr-list*), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**SR parameters**

SRs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
</table>
uuid | The unique identifier/object reference for the SR | read only |
<p>| name-label | The name of the SR | read/write |
| name-description | The description string of the SR | read/write |
| allowed-operations | list of the operations allowed on the SR in this state | read only set parameter |
| current-operations | A list of the operations that are currently in progress on this SR | read only set parameter |
| VDIs | unique identifier/object reference for the virtual disks in this SR | read only set parameter |
| PBDs | unique identifier/object reference for the PBDs attached to this SR | read only set parameter |
| physical-utilisation | physical space currently utilized on this SR, in bytes. Note that for sparse disk formats, physical utilization may be less than virtual allocation | read only |
| physical-size | total physical size of the SR, in bytes | read only |
| type | type of the SR, used to specify the SR backend driver to use | read only |
| content-type | the type of the SR's content. Currently used only to distinguish ISO libraries from other SRs. For storage repositories that store a library of ISOs, the <em>content-type</em> must be set to <em>iso</em>. In other cases, it is recommended that this be set either to empty, or the string 'user’. | read only |</p>
<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>shared</td>
<td>true if this SR is capable of being shared between multiple XenServer hosts; false otherwise</td>
<td>read/write</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters for the SR</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>host</td>
<td>The storage repository host name</td>
<td>read only</td>
</tr>
<tr>
<td>virtual-allocation</td>
<td>sum of virtual_sizes of all VDIs in this storage repository (in bytes)</td>
<td>read only</td>
</tr>
<tr>
<td>sm-config</td>
<td>SM dependent data</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>blobs</td>
<td>Binary data store</td>
<td>read only</td>
</tr>
</tbody>
</table>

**sr-create**

```
sr-create name-label=<name> physical-size=<size> type=<type> content-type=<content_type>
device-config:config_name=<value> [host-uuid=<XenServer host UUID>] [shared=true | false]
```

Makes an SR on the disk, introduces it into the database, and creates a PBD attaching the SR to a XenServer host. If `shared` is set to `true`, a PBD is created for each XenServer host in the pool; if `shared` is not specified or set to `false`, a PBD is created only for the XenServer host specified with `host-uuid`.

The exact `device-config` parameters differ depending on the device `type`. See the section called “Storage” for details of these parameters across the different storage backends.

**sr-destroy**

```
sr-destroy uuid=<sr_uuid>
```

Destroys the specified SR on the XenServer host.

**sr-forget**

```
sr-forget uuid=<sr_uuid>
```

The `xapi` agent forgets about a specified SR on the XenServer host, meaning that the SR is detached and you cannot access VDIs on it, but it remains intact on the source media (the data is not lost).

**sr-introduce**

```
sr-introduce name-label=<name> physical-size=<physical_size> type=<type> content-type=<content_type>
uuid=<sr_uuid>
```

Just places an SR record into the database. The device-config parameters are specified by `device-config:parameter_key=parameter_value` (for example, `device-config:device=/dev/sdb1`).
Note

This command is never used in normal operation. It is an advanced operation which might be useful if an SR needs to be reconfigured as shared after the fact, or to help recover from various failure scenarios.

sr-probe

**sr-probe** *type=* `<type>` [host-uuid=`<uuid_of_host>`] [device-config:`<config_name>`=`<value>`]

Performs a backend-specific scan, using the provided `device-config` keys. If the `device-config` is complete for the SR backend, then this will return a list of the SRs present on the device, if any. If the `device-config` parameters are only partial, then a backend-specific scan will be performed, returning results that will guide the user in improving the remaining `device-config` parameters. The scan results are returned as backend-specific XML, printed out by the CLI.

The exact `device-config` parameters differ depending on the device `type`. See the section called “Storage” for details of these parameters across the different storage backends.

sr-scan

**sr-scan** *uuid=* `<sr_uuid>`

Force an SR scan, syncing the xapi database with VDIs present in the underlying storage substrate.

Task commands

Commands for working with long-running asynchronous tasks. These are tasks such as starting, stopping, and suspending a Virtual Machine, which are typically made up of a set of other atomic subtasks that together accomplish the requested operation.

The task objects can be listed with the standard object listing command (**xe task-list**), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

Task parameters

Tasks have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the Task</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the Task</td>
<td>read only</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the Task</td>
<td>read only</td>
</tr>
<tr>
<td>resident-on</td>
<td>The unique identifier/object reference of the host on which the task is running</td>
<td>read only</td>
</tr>
<tr>
<td>status</td>
<td>current status of the Task</td>
<td>read only</td>
</tr>
<tr>
<td>progress</td>
<td>if the Task is still pending, this field contains the estimated percentage complete, from 0 to 100</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>1. If the Task has completed, successfully or unsuccessfully, this should be 1.</td>
<td>type</td>
<td>if the Task has completed successfully, this parameter contains the type of the encoded result - that is, the name of the class whose reference is in the result field; otherwise, this parameter's value is undefined</td>
</tr>
<tr>
<td>if the Task has completed successfully, this parameter contains the type of the encoded result - that is, the name of the class whose reference is in the result field; otherwise, this parameter's value is undefined</td>
<td>result</td>
<td>if the Task has completed successfully, this field contains the result value, either Void or an object reference; otherwise, this parameter's value is undefined</td>
</tr>
<tr>
<td>if the Task has failed, this parameter contains the set of associated error strings; otherwise, this parameter's value is undefined</td>
<td>error_info</td>
<td>read only</td>
</tr>
<tr>
<td>list of the operations allowed in this state</td>
<td>allowed_operations</td>
<td>read only</td>
</tr>
<tr>
<td>Time the task has been created</td>
<td>created</td>
<td>read only</td>
</tr>
<tr>
<td>Time task finished (i.e. succeeded or failed). If task-status is pending, then the value of this field has no meaning</td>
<td>finished</td>
<td>read only</td>
</tr>
<tr>
<td>Contains the UUID of the tasks this task is a sub-task of</td>
<td>subtask_of</td>
<td>read only</td>
</tr>
<tr>
<td>Contains the UUID(s) of all the subtasks of this task</td>
<td>subtasks</td>
<td>read only</td>
</tr>
</tbody>
</table>

### task-cancel

```
task-cancel [uuid=<task_uuid>]
```

Direct the specified Task to cancel and return.

### Template commands

Commands for working with VM templates.

Templates are essentially VMs with the *is-a-template* parameter set to *true*. A template is a "gold image" that contains all the various configuration settings to instantiate a specific VM. XenServer ships with a base set of templates, which range from generic "raw" VMs that can boot an OS vendor installation CD (RHEL, CentOS, SLES, Windows) to complete pre-configured OS instances (Debian Etch and Sarge). With XenServer you can create VMs, configure them in standard forms for your particular needs, and save a copy of them as templates for future use in VM deployment.

The template objects can be listed with the standard object listing command (*xe template-list*), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.
## Template parameters

Templates have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the template</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the template</td>
<td>read/write</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the template</td>
<td>read/write</td>
</tr>
<tr>
<td>user-version</td>
<td>string for creators of VMs and templates to put version information</td>
<td>read/write</td>
</tr>
<tr>
<td>is-a-template</td>
<td>true if this is a template. Template VMs can never be started, they are used only for cloning other VMs</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>Note that setting is-a-template using the CLI is not supported.</td>
<td></td>
</tr>
<tr>
<td>is-control-domain</td>
<td>true if this is a control domain (domain 0 or a driver domain)</td>
<td>read only</td>
</tr>
<tr>
<td>power-state</td>
<td>Current power state; always halted for a template</td>
<td>read only</td>
</tr>
<tr>
<td>memory-dynamic-max</td>
<td>dynamic maximum memory in bytes. Currently unused, but if changed the following constraint must be obeyed: memory_static_max &gt;= memory_dynamic_max &gt;= memory_dynamic_min &gt;= memory_static_min</td>
<td>read/write</td>
</tr>
<tr>
<td>memory-dynamic-min</td>
<td>dynamic minimum memory in bytes. Currently unused, but if changed the same constraints for memory-dynamic-max must be obeyed.</td>
<td>read/write</td>
</tr>
<tr>
<td>memory-static-max</td>
<td>statically-set (absolute) maximum memory in bytes. This is the main value used to determine the amount of memory assigned to a VM.</td>
<td>read/write</td>
</tr>
<tr>
<td>memory-static-min</td>
<td>statically-set (absolute) minimum memory in bytes. This represents the absolute minimum memory, and memory-static-min must be less than memory-static-max. This</td>
<td>read/write</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>suspend-VDI-uuid</td>
<td>The VDI that a suspend image is stored on (has no meaning for a template)</td>
<td>read only</td>
</tr>
<tr>
<td>VCPUs-params</td>
<td>configuration parameters for the selected VCPU policy.</td>
<td>read/write map parameter</td>
</tr>
</tbody>
</table>

You can tune a VCPU's pinning with

```
xe vm-param-set uuid=<vm_uuid> \ 
VCPUs-params:mask=1,2,3
```

A VM created from this template will then run on physical CPUs 1, 2, and 3 only.

You can also tune the VCPU priority (xen scheduling) with the `cap` and `weight` parameters; for example

```
xe vm-param-set uuid=<vm_uuid> \ 
VCPUs-params:weight=512
xe vm-param-set uuid=<vm_uuid> \ 
VCPUs-params:cap=100
```

A VM based on this template with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended XenServer host. Legal weights range from 1 to 65535 and the default is 256.

The cap optionally fixes the maximum amount of CPU a VM based on this template will be able to consume, even if the XenServer host has idle CPU cycles. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc. The default, 0, means there is no upper cap.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCPUs-max</td>
<td>Maximum number of VCPUs</td>
<td>read/write</td>
</tr>
<tr>
<td>VCPUs-at-startup</td>
<td>Boot number of VCPUs</td>
<td>read/write</td>
</tr>
<tr>
<td>actions-after-crash</td>
<td>action to take if a VM based on this template crashes</td>
<td>read/write</td>
</tr>
<tr>
<td>console-uuuids</td>
<td>virtual console devices</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>platform</td>
<td>platform-specific configuration</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>list of the operations allowed in this state</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>current-operations</td>
<td>A list of the operations that are currently in progress on this template</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>allowed-VBD-devices</td>
<td>list of VBD identifiers available for use, represented by integers of the range 0-15. This list is informational only, and other devices may be used (but may not work).</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>allowed-VIF-devices</td>
<td>list of VIF identifiers available for use, represented by integers of the range 0-15. This list is informational only, and other devices may be used (but may not work).</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>HVM-boot-policy</td>
<td></td>
<td>read/write</td>
</tr>
<tr>
<td>HVM-boot-params</td>
<td>The <code>order</code> key controls the HVM guest boot order, represented as a string where each character is a boot method: &quot;d&quot; for the CD/DVD, &quot;c&quot; for the root disk, and &quot;n&quot; for network PXE boot. The default is &quot;dc&quot;.</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>PV-kernel</td>
<td>path to the kernel</td>
<td>read/write</td>
</tr>
<tr>
<td>PV-ramdisk</td>
<td>path to the initrd</td>
<td>read/write</td>
</tr>
<tr>
<td>PV-args</td>
<td>string of kernel command line arguments</td>
<td>read/write</td>
</tr>
<tr>
<td>PV-legacy-args</td>
<td>string of arguments to make legacy VMs based on this template boot</td>
<td>read/write</td>
</tr>
<tr>
<td>PV-bootloader</td>
<td>name of or path to bootloader</td>
<td>read/write</td>
</tr>
<tr>
<td>PV-bootloader-args</td>
<td>string of miscellaneous arguments for the bootloader</td>
<td>read/write</td>
</tr>
<tr>
<td>last-boot-CPU-flags</td>
<td>describes the CPU flags on which a VM based on this template was last booted; not populated for a template</td>
<td>read only</td>
</tr>
<tr>
<td>resident-on</td>
<td>the XenServer host on which a VM based on this template is currently resident; appears as &lt;not in database&gt; for a template</td>
<td>read only</td>
</tr>
<tr>
<td>affinity</td>
<td>a XenServer host which a VM based on this template has preference for running on; used by the <code>xe vm-start</code> command to decide where to run the VM</td>
<td>read/write</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters for the template</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>start-time</td>
<td>Timestamp of the date and time that the metrics for a VM based on this template were</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>install-time</td>
<td>Timestamp of the date and time that the metrics for a VM based on this template were read, in the form ( \text{yyyymmddThh:mm:ss z} ), where ( z ) is the single-letter military timezone indicator, for example, ( Z ) for UTC (GMT); set to 1 Jan 1970 Z (beginning of Unix/POSIX epoch) for a template</td>
<td>read only</td>
</tr>
<tr>
<td>memory-actual</td>
<td>The actual memory being used by a VM based on this template; 0 for a template</td>
<td>read only</td>
</tr>
<tr>
<td>VCPUs-number</td>
<td>The number of virtual CPUs assigned to a VM based on this template; 0 for a template</td>
<td>read only</td>
</tr>
<tr>
<td>VCPUs-utilisation</td>
<td>A list of virtual CPUs and their weight</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>os-version</td>
<td>the version of the operating system for a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>PV-drivers-version</td>
<td>the versions of the paravirtualized drivers for a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>PV-drivers-up-to-date</td>
<td>flag for latest version of the paravirtualized drivers for a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only</td>
</tr>
<tr>
<td>memory</td>
<td>memory metrics reported by the agent on a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>disks</td>
<td>disk metrics reported by the agent on a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>networks</td>
<td>network metrics reported by the agent on a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>other</td>
<td>other metrics reported by the agent on a VM based on this template; appears as &lt;not in database&gt; for a template</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>guest-metrics-last-updated</td>
<td>timestamp when the last write to these fields was performed by the in-guest agent, in the</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>actions-after-shutdown</td>
<td>action to take after the VM has shutdown</td>
<td>read/write</td>
</tr>
<tr>
<td>actions-after-reboot</td>
<td>action to take after the VM has rebooted</td>
<td>read/write</td>
</tr>
<tr>
<td>possible-hosts</td>
<td>list of hosts that could potentially host the VM</td>
<td>read only</td>
</tr>
<tr>
<td>HVM-shadow-multiplier</td>
<td>multiplier applied to the amount of shadow that will be made available to the guest</td>
<td>read/write</td>
</tr>
<tr>
<td>dom-id</td>
<td>domain ID (if available, -1 otherwise)</td>
<td>read only</td>
</tr>
<tr>
<td>recommendations</td>
<td>An XML specification of recommended values and ranges for properties of this VM</td>
<td>read only</td>
</tr>
<tr>
<td>xenstore-data</td>
<td>data to be inserted into the xenstore tree (/local/domain/&lt;domid&gt;/vm-data) after the VM is created.</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>is-a-snapshot</td>
<td>True if this template is a VM snapshot</td>
<td>read only</td>
</tr>
<tr>
<td>snapshot_of</td>
<td>The UUID of the VM that this template is a snapshot of</td>
<td>read only</td>
</tr>
<tr>
<td>snapshots</td>
<td>The UUID(s) of any snapshots that have been taken of this template</td>
<td>read only</td>
</tr>
<tr>
<td>snapshot_time</td>
<td>The timestamp of the most recent VM snapshot taken</td>
<td>read only</td>
</tr>
<tr>
<td>memory-target</td>
<td>The target amount of memory set for this template</td>
<td>read only</td>
</tr>
<tr>
<td>blocked-operations</td>
<td>Lists the operations that cannot be performed on this template</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>last-boot-record</td>
<td>Record of the last boot parameters for this template, in XML format</td>
<td>read only</td>
</tr>
<tr>
<td>ha-always-run</td>
<td>True if an instance of this template will always restarted on another host in case of the failure of the host it is resident on</td>
<td>read/write</td>
</tr>
<tr>
<td>ha-restart-priority</td>
<td>1, 2, 3 or best effort. 1 is the highest restart priority</td>
<td>read/write</td>
</tr>
<tr>
<td>blobs</td>
<td>Binary data store</td>
<td>read only</td>
</tr>
<tr>
<td>live</td>
<td>Only relevant to a running VM.</td>
<td>read only</td>
</tr>
<tr>
<td>form</td>
<td>yyyyMMddThh:mm:ss z, where z is the single-letter military timezone indicator, for example, Z for UTC (GMT)</td>
<td></td>
</tr>
</tbody>
</table>
template-export

template-export template-uuid=<uuid_of_existing_template> filename=<filename_for_new_template>

Exports a copy of a specified template to a file with the specified new filename.

Update commands

Commands for working with updates to the OEM edition of XenServer. For commands relating to updating the standard non-OEM editions of XenServer, see the section called “Patch (update) commands” for details.

update-upload

update-upload file-name=<name_of_upload_file>

Streams a new software image to a OEM edition XenServer host. You must then restart the host for this to take effect.

User commands

user-password-change

user-password-change old=<old_password> new=<new_password>

Changes the logged-in user's password. The old password field is not currently checked because you require supervisor privilege to make this call.

VBD commands

Commands for working with VBDs (Virtual Block Devices).

A VBD is a software object that connects a VM to the VDI, which represents the contents of the virtual disk. The VBD has the attributes which tie the VDI to the VM (is it bootable, its read/write metrics, and so on), while the VDI has the information on the physical attributes of the virtual disk (which type of SR, whether the disk is shareable, whether the media is read/write or read only, and so on).

The VBD objects can be listed with the standard object listing command (xe vbd-list), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

VBD parameters

VBDs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the VBD</td>
<td>read only</td>
</tr>
<tr>
<td>vm-uuid</td>
<td>The unique identifier/object reference for the VM this VBD is attached to</td>
<td>read only</td>
</tr>
<tr>
<td>vm-name-label</td>
<td>The name of the VM this VBD is attached to</td>
<td>read only</td>
</tr>
<tr>
<td>vdi-uuid</td>
<td>The unique identifier/object reference for the VDI this VBD is mapped to</td>
<td>read only</td>
</tr>
<tr>
<td>vdi-name-label</td>
<td>The name of the VDI this VBD is mapped to</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>empty</td>
<td>if true, this represents an empty drive</td>
<td>read only</td>
</tr>
<tr>
<td>device</td>
<td>the device seen by the guest, for example hda1</td>
<td>read only</td>
</tr>
<tr>
<td>userdevice</td>
<td>user-friendly device name</td>
<td>read/write</td>
</tr>
<tr>
<td>bootable</td>
<td>true if this VBD is bootable</td>
<td>read/write</td>
</tr>
<tr>
<td>mode</td>
<td>the mode the VBD should be mounted with</td>
<td>read/write</td>
</tr>
<tr>
<td>type</td>
<td>how the VBD appears to the VM, for example disk or CD</td>
<td>read/write</td>
</tr>
<tr>
<td>currently-attached</td>
<td>Is the VBD currently attached on this host? true or false</td>
<td>read only</td>
</tr>
<tr>
<td>storage-lock</td>
<td>true if a storage-level lock was acquired</td>
<td>read only</td>
</tr>
<tr>
<td>status-code</td>
<td>error/success code associated with the last attach operation</td>
<td>read only</td>
</tr>
<tr>
<td>status-detail</td>
<td>error/success information associated with the last attach operation status</td>
<td>read only</td>
</tr>
<tr>
<td>qos_algorithm_type</td>
<td>the QoS algorithm to use</td>
<td>read/write</td>
</tr>
<tr>
<td>qos_algorithm_params</td>
<td>parameters for the chosen QoS algorithm</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>qos_supported_algorithms</td>
<td>supported QoS algorithms for this VBD</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>io_read_kbs</td>
<td>average read rate in kB per second for this VBD</td>
<td>read only</td>
</tr>
<tr>
<td>io_write_kbs</td>
<td>average write rate in kB per second for this VBD</td>
<td>read only</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>list of the operations allowed in this state. This list is advisory only and the server state may have changed by the time this field is read by a client.</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>current-operations</td>
<td>links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>unpluggable</td>
<td>true if this VBD will support hot-unplug</td>
<td>read/write</td>
</tr>
<tr>
<td>attachable</td>
<td>true if the device can be attached</td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>additional configuration</td>
<td>read/write map parameter</td>
</tr>
</tbody>
</table>
vbd-create

vbd-create vm-uuid= <uuid_of_the_vm> device= <device_value> 
vdi-uuid= <uuid_of_the_vdi_the_vbd_will_connect_to> [bootable=true] [type= <Disk | CD>] [mode= <RW | RO>]

Create a new VBD on a VM.

Appropriate values for the device field are listed in the parameter allowed-VBD-devices on the specified VM. Before any VBDs exist there, the allowable values are integers from 0-15.

If the type is Disk, vdi-uuid is required. Mode can be RO or RW for a Disk.

If the type is CD, vdi-uuid is optional; if no VDI is specified, an empty VBD will be created for the CD. Mode must be RO for a CD.

vbd-destroy

vbd-destroy uuid= <uuid_of_vbd>

Destroy the specified VBD.

If the VBD has its other-config:owner parameter set to true, the associated VDI will also be destroyed.

vbd-eject

vbd-eject uuid= <uuid_of_vbd>

Remove the media from the drive represented by a VBD. This command only works if the media is of a removable type (a physical CD or an ISO); otherwise an error message "VBD_NOT_REMOVABLE_MEDIA" is returned.

vbd-insert

vbd-insert uuid= <uuid_of_vbd> vdi-uuid= <uuid_of_vdi_containing_media>

Insert new media into the drive represented by a VBD. This command only works if the media is of a removable type (a physical CD or an ISO); otherwise an error message "VBD_NOT_REMOVABLE_MEDIA" is returned.

vbd-plug

vbd-plug uuid= <uuid_of_vbd>

Attempt to attach the VBD while the VM is in the running state.

vbd-unplug

vbd-unplug uuid= <uuid_of_vbd>

Attempts to detach the VBD from the VM while it is in the running state.

VDI commands

Commands for working with VDIs (Virtual Disk Images).

A VDI is a software object that represents the contents of the virtual disk seen by a VM, as opposed to the VBD, which is a connector object that ties a VM to the VDI. The VDI has the information on the physical attributes of the virtual disk (which type of SR, whether the disk is shareable, whether the media is read/write or read only, and so on), while the VBD has the attributes which tie the VDI to the VM (is it bootable, its read/write metrics, and so on).
The VDI objects can be listed with the standard object listing command (`xe vdi-list`), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**VDI parameters**

VDIs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the VDI</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the VDI</td>
<td>read/write</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the VDI</td>
<td>read/write</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>a list of the operations allowed in this state</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>current-operations</td>
<td>a list of the operations that are currently in progress on this VDI</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>sr-uuid</td>
<td>SR in which the VDI resides</td>
<td>read only</td>
</tr>
<tr>
<td>vbd-uuids</td>
<td>a list of VBDs that refer to this VDI</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>crashdump-uuids</td>
<td>list of crash dumps that refer to this VDI</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>virtual-size</td>
<td>size of disk as presented to the VM, in bytes. Note that, depending on storage backend type, the size may not be respected exactly</td>
<td>read only</td>
</tr>
<tr>
<td>physical-utilisation</td>
<td>amount of physical space that the VDI is currently taking up on the SR, in bytes</td>
<td>read only</td>
</tr>
<tr>
<td>type</td>
<td>type of VDI, for example, System or User</td>
<td>read only</td>
</tr>
<tr>
<td>sharable</td>
<td>true if this VDI may be shared</td>
<td>read only</td>
</tr>
<tr>
<td>read-only</td>
<td>true if this VDI can only be mounted read-only</td>
<td>read only</td>
</tr>
<tr>
<td>storage-lock</td>
<td>true if this VDI is locked at the storage level</td>
<td>read only</td>
</tr>
<tr>
<td>parent</td>
<td>References the parent VDI, if this VDI is part of a chain</td>
<td>read only</td>
</tr>
<tr>
<td>missing</td>
<td>true if SR scan operation reported this VDI as not present</td>
<td>read only</td>
</tr>
<tr>
<td>other-config</td>
<td>additional configuration information for this VDI</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>sr-name-label</td>
<td>name of the containing storage repository</td>
<td>read only</td>
</tr>
<tr>
<td>location</td>
<td>location information</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>managed</td>
<td>true if the CDI is managed</td>
<td>read only</td>
</tr>
<tr>
<td>xenstore-data</td>
<td>data to be inserted into the xenstore tree (/local/domain/0/backend/vbd/&lt;domid&gt;/&lt;device-id&gt;/sm-data) after the VDI is attached. This is generally set by the SM backends on vdi_attach.</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>sm-config</td>
<td>SM dependent data</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>is-a-snapshot</td>
<td>True if this VDI is a VM storage snapshot</td>
<td>read only</td>
</tr>
<tr>
<td>snapshot_of</td>
<td>The UUID of the storage this VDI is a snapshot of</td>
<td>read only</td>
</tr>
<tr>
<td>snapshots</td>
<td>The UUID(s) of all snapshots of this VDI</td>
<td>read only</td>
</tr>
<tr>
<td>snapshot_time</td>
<td>The timestamp of the snapshot operation that created this VDI</td>
<td>read only</td>
</tr>
</tbody>
</table>

**vdi-clone**

```
vdi-clone uuid=<uuid_of_the_vdi> [driver-params:<key=value>]```

Produce a new, writable copy of the specified VDI that can be used directly. It is a variant of **vdi-copy** that is capable of exposing high-speed image clone facilities where they exist.

The optional `driver-params` map parameter can be used for passing extra vendor-specific configuration information to the back end storage driver that the VDI is based on. See the storage vendor's driver documentation for details.

**vdi-copy**

```
vdi-copy uuid=<uuid_of_the_vdi> sr-uuid=<uuid_of_the_sr_to_where_you_want_to_copy_the_vdi>```

Copy a VDI to a specified SR.

**vdi-create**

```
vdi-create sr-uuid=<uuid_of_the_sr_where_you_want_to_create_the_vdi>
name-label=<name_for_the_vdi> type=<system | user | suspend | crashdump> virtual-size=<size_of_virtual_disk>
sm-config-*=<storage_specific_configuration_data>```

Create a VDI.

The `virtual-size` parameter can be specified in bytes or using the IEC standard suffixes KiB ($2^{10}$ bytes), MiB ($2^{20}$ bytes), GiB ($2^{30}$ bytes), and TiB ($2^{40}$ bytes).

**Note**

SR types that support sparse allocation of disks (such as Local VHD and NFS) do not enforce virtual allocation of disks. Users should therefore take great care when over-allocating virtual disk space on an SR. If an over-allocated
SR does become full, disk space must be made available either on the SR target substrate or by deleting unused VDIs in the SR.

**Note**

Some SR types might round up the `virtual-size` value to make it divisible by a configured block size.

### vdi-destroy

```
vdi-destroy uuid=<uuid_of_vdi>
```

Destroy the specified VDI.

**Note**

In the case of Local VHD and NFS SR types, disk space is not immediately released on `vdi-destroy`, but periodically during a storage repository scan operation. Users that need to force deleted disk space to be made available should call `sr-scan` manually.

### vdi-forget

```
vdi-forget uuid=<uuid_of_vdi>
```

Unconditionally removes a VDI record from the database without touching the storage backend. In normal operation, you should be using `vdi-destroy` instead.

### vdi-import

```
vdi-import uuid=<uuid_of_vdi> filename=<filename_of_raw_vdi>
```

Import a raw VDI.

### vdi-introduce

```
vdi-introduce uuid=<uuid_of_vdi> sr-uuid=<uuid_of_sr_to_import_into>
    name-label=<name_of_the_new_vdi>
    type=<system | user | suspend | crashdump> location=<device_location_(varies_by_storage_type)>
    [name-description=<description_of_vdi>]
    [sharable=<yes | no>]
    [read-only=<yes | no>]
    [other-config=<map_to_store_misuser_specific_data>]
    [xenstore-data=<map_to_of_additional_xenstore_keys>]
    [sm-config<storage_specific_configuration_data>]
```

Create a VDI object representing an existing storage device, without actually modifying or creating any storage. This command is primarily used internally to automatically introduce hot-plugged storage devices.
### vdi-resize

vdi-resize uuid=\(<vdi\_uuid>\) disk-size=\(<new\_size\_for\_disk>\)

Resize the VDI specified by UUID.

### vdi-snapshot

vdi-snapshot uuid=\(<uuid\_of\_the\_vdi>\) [driver-params=\(<params>\)]

Produces a read-write version of a VDI that can be used as a reference for backup and/or templating purposes. You can perform a backup from a snapshot rather than installing and running backup software inside the VM. The VM can continue running while external backup software streams the contents of the snapshot to the backup media. Similarly, a snapshot can be used as a "gold image" on which to base a template. A template can be made using any VDIs.

The optional `driver-params` map parameter can be used for passing extra vendor-specific configuration information to the back end storage driver that the VDI is based on. See the storage vendor’s driver documentation for details.

A clone of a snapshot should always produce a writable VDI.

### vdi-unlock

vdi-unlock uuid=\(<uuid\_of\_vdi\_to\_unlock>\) [force=true]

Attempts to unlock the specified VDIs. If `force=true` is passed to the command, it will force the unlocking operation.

### VIF commands

Commands for working with VIFs (Virtual network interfaces).

The VIF objects can be listed with the standard object listing command (xe vif-list), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

### VIF parameters

VIFs have the following parameters:

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the VIF</td>
<td>read only</td>
</tr>
<tr>
<td>vm-uuid</td>
<td>The unique identifier/object reference for the VM that this VIF resides on</td>
<td>read only</td>
</tr>
<tr>
<td>vm-name-label</td>
<td>The name of the VM that this VIF resides on</td>
<td>read only</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>a list of the operations allowed in this state</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>current-operations</td>
<td>a list of the operations that are currently in progress on this VIF</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>device</td>
<td>integer label of this VIF, indicating the order in which VIF backends were created</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>MAC</td>
<td>MAC address of VIF, as exposed to the VM</td>
<td>read only</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum Transmission Unit of the VIF in bytes. This parameter is read-only,</td>
<td>read only</td>
</tr>
<tr>
<td></td>
<td>but you can override the MTU setting with the ( mtu ) key via the (</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other-config ) map parameter. For example, to reset the MTU on a virtual</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NIC to use jumbo frames:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>`xe vif-param-set uuid=&lt;vif_uuid&gt; \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>other-config:mtu=9000`</td>
<td></td>
</tr>
<tr>
<td>currently-attached</td>
<td>true if the device is currently attached</td>
<td>read only</td>
</tr>
<tr>
<td>qos_algorithm_type</td>
<td>QoS algorithm to use</td>
<td>read/write</td>
</tr>
<tr>
<td>qos_algorithm_params</td>
<td>parameters for the chosen QoS algorithm</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>qos_supported_algorithms</td>
<td>supported QoS algorithms for this VIF</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>other-config</td>
<td>A list of key/value pairs that specify additional configuration parameters</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td></td>
<td>for this VIF.</td>
<td></td>
</tr>
<tr>
<td>network-uuid</td>
<td>The unique identifier/object reference of the virtual network to which this</td>
<td>read only</td>
</tr>
<tr>
<td></td>
<td>VIF is connected</td>
<td></td>
</tr>
<tr>
<td>network-name-label</td>
<td>The descriptive name of the virtual network to which this VIF is connected</td>
<td>read only</td>
</tr>
<tr>
<td>io_read_kbs</td>
<td>average read rate in kB/s for this VIF</td>
<td>read only</td>
</tr>
<tr>
<td>io_write_kbs</td>
<td>average write rate in kB/s for this VIF</td>
<td>read only</td>
</tr>
</tbody>
</table>

**vif-create**

vif-create vm-uuid=<uuid_of_the_vm> device= <see below> 
network-uuid= <uuid_of_the_network_the_vif_will_connect_to> [mac=<mac_address>]

Create a new VIF on a VM.

Appropriate values for the `device` field are listed in the parameter `allowed-VIF-devices` on the specified VM. Before any VIFs exist there, the allowable values are integers from 0-15.

The `mac` parameter is the standard MAC address in the form `aa:bb:cc:dd:ee:ff`. If you leave it unspecified, an appropriate random MAC address will be created. You can also explicitly set a random MAC address by specifying `mac=random`.

**vif-destroy**

vif-destroy uuid=<uuid_of_vif>

Destroy a VIF.
**vif-plug**

vif-plug uuid=\(<uuid\_of\_vif>\)

Attempt to attach the VIF while the VM is in the running state.

**vif-unplug**

vif-unplug uuid=\(<uuid\_of\_vif>\)

Attempts to detach the VIF from the VM while it is in the running state.

**VLAN commands**

Commands for working with VLANs (virtual networks). To list and edit virtual interfaces, refer to the PIF commands, which have a VLAN parameter to signal that they have an associated virtual network (see the section called “PIF commands”). For example, to list VLANs you need to use `xe pif-list`.

**vlan-create**

vlan-create pif-uuid=\(<uuid\_of\_pif>\) vlan=\(<vlan\_tag>\) network-uuid=\(<uuid\_of\_network>\)

Create a new VLAN on a XenServer host.

**vlan-destroy**

vlan-destroy uuid=\(<uuid\_of\_pif\_mapped\_to\_vlan>\)

Destroy a VLAN. Requires the UUID of the PIF that represents the VLAN.

**VM commands**

Commands for controlling VMs and their attributes.

**VM selectors**

Several of the commands listed here have a common mechanism for selecting one or more VMs on which to perform the operation. The simplest way is by supplying the argument `vm=\(<name\_or\_uuid>\)`. VMs can also be specified by filtering the full list of VMs on the values of fields. For example, specifying `power-state=halted` will select all VMs whose `power-state` parameter is equal to `halted`. Where multiple VMs are matching, the option `--multiple` must be specified to perform the operation. The full list of parameters that can be matched is described at the beginning of this section, and can be obtained by the command `xe vm-list params=all`. If no parameters to select VMs are given, the operation will be performed on all VMs.

The VM objects can be listed with the standard object listing command (`xe vm-list`), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

**VM parameters**

VMs have the following parameters:
**Note**

All writeable VM parameter values can be changed while the VM is running, but the new parameters are *not* applied dynamically and will not be applied until the VM is rebooted.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>uuid</td>
<td>The unique identifier/object reference for the VM</td>
<td>read only</td>
</tr>
<tr>
<td>name-label</td>
<td>The name of the VM</td>
<td>read/write</td>
</tr>
<tr>
<td>name-description</td>
<td>The description string of the VM</td>
<td>read/write</td>
</tr>
<tr>
<td>user-version</td>
<td>string for creators of VMs and templates to put version information</td>
<td>read/write</td>
</tr>
<tr>
<td>is-a-template</td>
<td>false unless this is a template; template VMs can never be started, they are used only for cloning other VMs</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>Note that setting <em>is-a-template</em> using the CLI is not supported.</td>
<td></td>
</tr>
<tr>
<td>is-control-domain</td>
<td>true if this is a control domain (domain 0 or a driver domain)</td>
<td>read only</td>
</tr>
<tr>
<td>power-state</td>
<td>Current power state</td>
<td>read only</td>
</tr>
<tr>
<td>memory-dynamic-max</td>
<td>dynamic maximum in bytes</td>
<td>read/write</td>
</tr>
<tr>
<td>memory-dynamic-min</td>
<td>dynamic minimum in bytes</td>
<td>read/write</td>
</tr>
<tr>
<td>memory-static-max</td>
<td>statically-set (absolute) maximum in bytes.</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>If you want to change this value, the VM must be shut down.</td>
<td></td>
</tr>
<tr>
<td>memory-static-min</td>
<td>statically-set (absolute) minimum in bytes.</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>If you want to change this value, the VM must be shut down.</td>
<td></td>
</tr>
<tr>
<td>suspend-VDI-uuid</td>
<td>The VDI that a suspend image is stored on</td>
<td>read only</td>
</tr>
<tr>
<td>VCPUs-params</td>
<td>configuration parameters for the selected VCPU policy.</td>
<td>read/write</td>
</tr>
<tr>
<td></td>
<td>You can tune a VCPU's pinning with</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>xe vm-param-set uuid=&lt;vm_uuid&gt; \ VCPUs-params:mask=1,2,3</code></td>
<td></td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td></td>
<td>The selected VM will then run on physical CPUs 1, 2, and 3 only.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can also tune the VCPU priority (xen scheduling) with the <em>cap</em> and <em>weight</em> parameters; for example</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xe vm-param-set \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>uuid=&lt;template_uuid&gt; \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VCPUs-params:weight=512 \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xe vm-param-set \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>uuid=&lt;template UUID&gt; \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VCPUs-params:cap=100 \</td>
<td></td>
</tr>
<tr>
<td></td>
<td>A VM with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended XenServer host. Legal weights range from 1 to 65535 and the default is 256.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The cap optionally fixes the maximum amount of CPU a VM will be able to consume, even if the XenServer host has idle CPU cycles. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc. The default, 0, means there is no upper cap.</td>
<td></td>
</tr>
<tr>
<td>VCPUs-max</td>
<td>Maximum number of virtual CPUs.</td>
<td>read/write</td>
</tr>
<tr>
<td>VCPUs-at-startup</td>
<td>Boot number of virtual CPUs</td>
<td>read/write</td>
</tr>
<tr>
<td>actions-after-crash</td>
<td>action to take if the VM crashes. For PV guests, valid parameters are: preserve (for analysis only), coredump_and_restart (record a coredump and reboot VM), coredump_and_destroy (record a coredump and leave VM halted), restart (no coredump and restart VM), and destroy (no coredump and leave VM halted).</td>
<td>read/write</td>
</tr>
<tr>
<td>console-uuids</td>
<td>virtual console devices</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>platform</td>
<td>platform-specific configuration</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>allowed-operations</td>
<td>list of the operations allowed in this state</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>current-operations</td>
<td>A list of the operations that are currently in progress on the VM</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>allowed-VBD-devices</td>
<td>list of VBD identifiers available for use, represented by integers of the range 0-15. This</td>
<td>read only set parameter</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>list is informational only, and other devices may be used (but may not work).</td>
<td>read only set parameter</td>
<td></td>
</tr>
<tr>
<td>list of VIF identifiers available for use, represented by integers of the range 0-15. This list is informational only, and other devices may be used (but may not work).</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>list of VIF identifiers available for use, represented by integers of the range 0-15. This list is informational only, and other devices may be used (but may not work).</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>Floating point value which controls the amount of shadow memory overhead to grant the VM. Defaults to 1.0 (the minimum value), and should only be changed by advanced users.</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>path to the kernel</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>path to the initrd</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>string of kernel command line arguments</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>string of arguments to make legacy VMs boot</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>name of or path to bootloader</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>string of miscellaneous arguments for the bootloader</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>describes the CPU flags on which the VM was last booted</td>
<td>read only</td>
<td></td>
</tr>
<tr>
<td>the XenServer host on which a VM is currently resident</td>
<td>read only</td>
<td></td>
</tr>
<tr>
<td>a XenServer host which the VM has preference for running on; used by the <code>xe vm-start</code> command to decide where to run the VM</td>
<td>read/write</td>
<td></td>
</tr>
<tr>
<td>A list of key/value pairs that specify additional configuration parameters for the VM</td>
<td>read/write map parameter</td>
<td></td>
</tr>
<tr>
<td>For example, a VM will be started automatically after host boot if the other-config parameter includes the key/value pair <code>auto_poweron: true</code></td>
<td>read/write map parameter</td>
<td></td>
</tr>
<tr>
<td>Timestamp of the date and time that the metrics for the VM were read, in the form <code>yyyyymmddTHh:mm:ss z</code>, where <code>z</code> is the sin-</td>
<td>read only</td>
<td></td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>gle-letter military timezone indicator, for example, Z for UTC (GMT)</td>
<td></td>
<td>read only</td>
</tr>
<tr>
<td>install-time</td>
<td>Timestamp of the date and time that the metrics for the VM were read, in the form <code>yyyyymmddThh:mm:ss z</code>, where <code>z</code> is the single-letter military timezone indicator, for example, Z for UTC (GMT)</td>
<td>read only</td>
</tr>
<tr>
<td>memory-actual</td>
<td>The actual memory being used by a VM</td>
<td>read only</td>
</tr>
<tr>
<td>VCPUs-number</td>
<td>The number of virtual CPUs assigned to the VM</td>
<td>read only</td>
</tr>
<tr>
<td></td>
<td>For a paravirtualized Linux VM, this number can differ from <code>VCPUs-max</code> and can be changed without rebooting the VM via the <code>vm-vcpu-hotplug</code> command. See the section called “vm-vcpu-hotplug”. Windows VMs always run with the number of vCPUs set to <code>VCPUs-max</code> and must be rebooted to change this value.</td>
<td>read only</td>
</tr>
<tr>
<td></td>
<td>Note that performance will drop sharply if you set <code>VCPUs-number</code> to a value greater than the number of physical CPUs on the XenServer host.</td>
<td>read only</td>
</tr>
<tr>
<td>VCPUs-utilisation</td>
<td>A list of virtual CPUs and their weight</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>os-version</td>
<td>the version of the operating system for the VM</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>PV-drivers-version</td>
<td>the versions of the paravirtualized drivers for the VM</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>PV-drivers-up-to-date</td>
<td>flag for latest version of the paravirtualized drivers for the VM</td>
<td>read only</td>
</tr>
<tr>
<td>memory</td>
<td>memory metrics reported by the agent on the VM</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>disks</td>
<td>disk metrics reported by the agent on the VM</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>networks</td>
<td>network metrics reported by the agent on the VM</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>other</td>
<td>other metrics reported by the agent on the VM</td>
<td>read only map parameter</td>
</tr>
<tr>
<td>guest-metrics-last-updated</td>
<td>timestamp when the last write to these fields was performed by the in-guest agent, in the form <code>yyyyymmddThh:mm:ss z</code>, where <code>z</code> is the single-letter military timezone indicator, for example, Z for UTC (GMT)</td>
<td>read only</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
<td>Type</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>single-letter military timezone indicator, for example, Z for UTC (GMT)</td>
<td>actions-after-shutdown action to take after the VM has shutdown</td>
<td>read/write</td>
</tr>
<tr>
<td>actions-after-reboot</td>
<td>action to take after the VM has rebooted</td>
<td>read/write</td>
</tr>
<tr>
<td>possible-hosts</td>
<td>potential hosts of this VM</td>
<td>read only</td>
</tr>
<tr>
<td>dom-id</td>
<td>domain ID (if available, -1 otherwise)</td>
<td>read only</td>
</tr>
<tr>
<td>recommendations</td>
<td>An XML specification of recommended values and ranges for properties of this VM</td>
<td>read only</td>
</tr>
<tr>
<td>xenstore-data</td>
<td>data to be inserted into the xenstore tree (/local/domain/&lt;domid&gt;/vm-data) after the VM is created</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>is-a-snapshot</td>
<td>True if this VM is a snapshot</td>
<td>read only</td>
</tr>
<tr>
<td>snapshot_of</td>
<td>The UUID of the VM this is a snapshot of</td>
<td>read only</td>
</tr>
<tr>
<td>snapshots</td>
<td>The UUID(s) of all snapshots of this VM</td>
<td>read only</td>
</tr>
<tr>
<td>snapshot_time</td>
<td>the timestamp of the snapshot operation that created this VM snapshot</td>
<td>read only</td>
</tr>
<tr>
<td>memory-target</td>
<td>The target amount of memory set for this VM</td>
<td>read only</td>
</tr>
<tr>
<td>blocked-operations</td>
<td>Lists the operations that cannot be performed on this VM</td>
<td>read/write map parameter</td>
</tr>
<tr>
<td>last-boot-record</td>
<td>Record of the last boot parameters for this template, in XML format</td>
<td>read only</td>
</tr>
<tr>
<td>ha-always-run</td>
<td>True if this VM will always restarted on another host in case of the failure of the host it is resident on</td>
<td>read/write</td>
</tr>
<tr>
<td>ha-restart-priority</td>
<td>1, 2, 3 or best effort. 1 is the highest restart priority</td>
<td>read/write</td>
</tr>
<tr>
<td>blobs</td>
<td>Binary data store</td>
<td>read only</td>
</tr>
<tr>
<td>live</td>
<td>True if the VM is running, false if HA suspects that the VM may not be running.</td>
<td>read only</td>
</tr>
</tbody>
</table>

**vm-cd-add**

```
vm-cd-add cd-name=<name_of_new_cd> device=<integer_value_of_an_available_vbd>
[<vm-selector>=<vm_selector_value>...]...
```

Add a new virtual CD to the selected VM. The `device` parameter should be selected from the value of the `allowed-VBD-devices` parameter of the VM.
The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-cd-eject**

```bash
vm-cd-eject [vm-selector]=<vm_selector_value>...]
```

Eject a CD from the virtual CD drive. This command will only work if there is one and only one CD attached to the VM. When there are two or more CDs, please use the command `xe vbd-eject` and specify the UUID of the VBD.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-cd-insert**

```bash
vm-cd-insert cd-name=<name_of_cd> [vm-selector]=<vm_selector_value>...]
```

Insert a CD into the virtual CD drive. This command will only work if there is one and only one empty CD device attached to the VM. When there are two or more empty CD devices, please use the command `xe vbd-insert` and specify the UUIDs of the VBD and of the VDI to insert.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-cd-list**

```bash
vm-cd-list [vbd-params] [vdi-params] [vm-selector]=<vm_selector_value>...]
```

Lists CDs attached to the specified VMs.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

You can also select which VBD and VDI parameters to list.

**vm-cd-remove**

```bash
vm-cd-remove cd-name=<name_of_cd> [vm-selector]=<vm_selector_value>...]
```

Remove a virtual CD from the specified VMs.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-clone**

```bash
vm-clone new-name-label=<name_for_clone>
[new-name-description=<description_for_clone>] [vm-selector]=<vm_selector_value>...]
```

Clone an existing VM, using storage-level fast disk clone operation where available. Specify the name and the optional description for the resulting cloned VM using the `new-name-label` and `new-name-description` arguments.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.
**vm-compute-maximum-memory**

`vm-compute-maximum-memory total=<$amount_of_available_physical_ram_in_bytes> [approximate=<$add_overhead_memory_for_additional_vCPUs\? true | false>] [<$vm_selector>=<$vm_selector_value>...]`

Calculate the maximum amount of static memory which can be allocated to an existing VM, using the total amount of physical RAM as an upper bound. The optional parameter `approximate` reserves sufficient extra memory in the calculation to account for adding extra vCPUs into the VM at a later date.

For example:

```bash
xe vm-compute-maximum-memory vm=testvm total=`xe host-list params=memory-free --minimal`
```

uses the value of the `memory-free` parameter returned by the `xe host-list` command to set the maximum memory of the VM named "testvm."

The VM or VMs on which this operation will be performed are selected via the standard selection mechanism (see **VM selectors**). Optional arguments can be any number of the **VM parameters** listed at the beginning of this section.

**vm-copy**

`vm-copy new-name-label=<$name_for_copy> [new-name-description=<$description_for_copy>] [sr-uuid=<$uuid_of_sr>] [<$vm-selector>=<$vm_selector_value>...]`

Copy an existing VM, but without using storage-level fast disk clone operation (even if this is available). The disk images of the copied VM are guaranteed to be "full images" - that is, not part of a copy-on-write (CoW) chain.

Specify the name and the optional description for the resulting copied VM using the `new-name-label` and `new-name-description` arguments.

Specify the destination SR for the resulting copied VM using the `sr-uuid`. If this parameter is not specified, the destination is the same SR that the original VM is in.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see **VM selectors**). Optional arguments can be any number of the **VM parameters** listed at the beginning of this section.

**vm-crashdump-list**

`vm-crashdump-list [<$vm-selector>=<$vm_selector_value>...]`

List crashdumps associated with the specified VMs.

If the optional argument `params` is used, the value of params is a string containing a list of parameters of this object that you want to display. Alternatively, you can use the keyword `all` to show all parameters. If `params` is not used, the returned list shows a default subset of all available parameters.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see **VM selectors**). Optional arguments can be any number of the **VM parameters** listed at the beginning of this section.

**vm-destroy**

`vm-destroy uuid=<$uuid_of_vm>`
Destroy the specified VM. This leaves the storage associated with the VM intact. To delete storage as well, use `xe vm-uninstall`.

**vm-disk-add**

`vm-disk-add disk-size=<size_of_disk_to_add> device= <uuid_of_device> [ <vm-selector>=<vm_selector_value>... ]`

Add a new disk to the specified VMs. The `device` parameter should be selected from the value of the `allowed-VBD-devices` parameter of the VMs.

The `disk-size` parameter can be specified in bytes or using the IEC standard suffixes KiB \( (2^{10} \) bytes), MiB \( (2^{20} \) bytes), GiB \( (2^{30} \) bytes), and TiB \( (2^{40} \) bytes).

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-disk-list**

`vm-disk-list [vbd-params] [vdi-params] [ <vm-selector>=<vm_selector_value>... ]`

Lists disks attached to the specified VMs. The `vbd-params` and `vdi-params` parameters control the fields of the respective objects to output and should be given as a comma-separated list, or the special key `all` for the complete list.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-disk-remove**

`vm-disk-remove device= <integer_label_of_disk> [ <vm-selector>=<vm_selector_value>... ]`

Remove a disk from the specified VMs and destroy it.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-export**

`vm-export filename= <export_filename> [metadata= <true | false>] [ <vm-selector>=<vm_selector_value>... ]`

Export the specified VMs (including disk images) to a file on the local machine. Specify the filename to export the VM into using the `filename` parameter. By convention, the filename should have a `.xva` extension.

If the `metadata` parameter is `true`, then the disks are not exported, and only the VM metadata is written to the output file. This is intended to be used when the underlying storage is transferred through other mechanisms, and permits the VM information to be recreated (see the section called “vm-import”).

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

**vm-import**

`vm-import filename= <export_filename> [metadata= <true | false>]`
[preserve=<true | false>]
[sr-uuid=<destination_sr_uuid>]

Import a VM from a previously-exported file. If `preserve` is set to `true`, the MAC address of the original VM will be preserved. The `sr-uuid` determines the destination SR to import the VM into, and is the default SR if not specified.

The `filename` parameter can also point to an XVA-format VM, which is the legacy export format from XenServer 3.2 and is used by some third-party vendors to provide virtual appliances. This format uses a directory to store the VM data, so set `filename` to the root directory of the XVA export and not an actual file. Subsequent exports of the imported legacy guest will automatically be upgraded to the new filename-based format, which stores much more data about the configuration of the VM.

**Note**

The older directory-based XVA format does not fully preserve all the VM attributes. In particular, imported VMs will not have any virtual network interfaces attached by default. If networking is required, create one using `vif-create` and `vif-plug`.

If the `metadata` is `true`, then a previously exported set of metadata can be imported without their associated disk blocks. Metadata-only import will fail if any VDIs cannot be found (named by SR and `VDI.location`) unless the `--force` option is specified, in which case the import will proceed regardless. If disks can be mirrored/moved out-of-band then metadata import/export represents a fast way of moving VMs between disjoint pools (e.g. as part of a disaster recovery plan).

### vm-install

```
vm-install new-name-label=<name>
[template-uuid=<uuid_of_desired_template> | [template=<uuid_or_name_of_desired_template>]]
[sr-uuid=<sr_uuid> | sr-name-label=<name_of_sr>]
```

Install a VM from a template. Specify the template name using either the `template-uuid` or `template` argument. Specify an SR other than the default SR using either the `sr-uuid` or `sr-name-label` argument.

### vm-memory-shadow-multiplier-set

```
vm-memory-shadow-multiplier-set [<vm-selector>=<vm_selector_value>...]
[multiplier=<float_memory_multiplier>]
```

Set the shadow memory multiplier for the specified VM.

This is an advanced option which modifies the amount of shadow memory assigned to a hardware-assisted VM. In some specialized application workloads, such as the Citrix Presentation Server, extra shadow memory is required to achieve full performance.

This memory is considered to be an overhead. It is separated from the normal memory calculations for accounting memory to a VM. When this command is invoked, the amount of free XenServer host memory will decrease according to the multiplier, and the `HVM_shadow_multiplier` field will be updated with the actual value which Xen has assigned to the VM. If there is not enough XenServer host memory free, then an error will be returned.

The VMs on which this operation should be performed are selected via the standard selection mechanism (see `VM selectors` above).
vm-migrate

vm-migrate [[host-uuid=<destination XenServer host UUID>] [host=<name or UUID of destination XenServer host>] [vm-selector=<vm_selector_value>...] [live=<true | false>]

Migrate the specified VMs between physical hosts. The host parameter can be either the name or the UUID of the XenServer host.

By default, the VM will be suspended, migrated, and resumed on the other host. The live parameter activates XenMotion and keeps the VM running while performing the migration, thus minimizing VM downtime to less than a second. In some circumstances such as extremely memory-heavy workloads in the VM, XenMotion automatically falls back into the default mode and suspends the VM for a brief period of time before completing the memory transfer.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-reboot

vm-reboot [vm-selector=<vm_selector_value>...] [force=<true>]

Reboot the specified VMs.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

Use the force argument to cause an ungraceful shutdown, akin to pulling the plug on a physical server.

vm-reset-powerstate

vm-reset-powerstate [vm-selector=<vm_selector_value>...] {force=true}

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

This is an advanced command only to be used when a member host in a pool goes down. You can use this command to force the pool master to reset the power-state of the VMs to be "halted". Essentially this forces the lock on the VM and its disks so it can be subsequently started on another pool host. This call requires the force flag to be specified, and fails if it is not on the command-line.

vm-resume

vm-resume [vm-selector=<vm_selector_value>...] [force=<true | false>] [on=<XenServer host UUID>]

Resume the specified VMs.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

If the VM is on a shared SR in a pool of hosts, use the on argument to specify which host in the pool on which to start it. By default the system will determine an appropriate host, which might be any of the members of the pool.

vm-shutdown

vm-shutdown [vm-selector=<vm_selector_value>...] [force=<true | false>]

Shut down the specified VM.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

Use the force argument to cause an ungraceful shutdown, akin to pulling the plug on a physical server.

```
vm-start
```

```
vm-start [ <vm-selector>=<vm_selector_value> ... ] [ force=<true | false> ] [ on=<XenServer host UUID> ] [--multiple]
```

Start the specified VMs.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

If the VMs are on a shared SR in a pool of hosts, use the on argument to specify which host in the pool on which to start the VMs. By default the system will determine an appropriate host, which might be any of the members of the pool.

```
vm-suspend
```

```
vm-suspend [ <vm-selector>=<vm_selector_value> ... ]
```

Suspend the specified VM.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

```
vm-uninstall
```

```
vm-uninstall [ <vm-selector>=<vm_selector_value> ... ] [ force=<true | false> ]
```

Uninstall a VM, destroying its disks (those VDIs that are marked RW and connected to this VM only) as well as its metadata record. To simply destroy the VM metadata, use `xe vm-destroy`.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

```
vm-vcpu-hotplug
```

```
vcpu-new-vcpus=<new_vcpu_count> [ <vm-selector>=<vm_selector_value> ... ]
```

Dynamically adjust the number of vCPUs available to a running paravirtual Linux VM within the number bounded by the parameter `VCPUs-max`. Windows VMs always run with the number of vCPUs set to `VCPUs-max` and must be rebooted to change this value.

The paravirtualized Linux VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

```
vm-vif-list
```

```
vif-list [ <vm-selector>=<vm_selector_value> ... ]
```


Lists the VIFs from the specified VMs.

The VM or VMs on which this operation should be performed are selected via the standard selection mechanism (see VM selectors). Note that the selectors operate on the VM records when filtering, and not on the VIF values. Optional arguments can be any number of the VM parameters listed at the beginning of this section.
Chapter 7. Troubleshooting

If you experience odd behavior, application crashes, or have other issues with a XenServer host, this chapter is meant to help you solve the problem if possible and, failing that, describes where the application logs are located and other information that can help your Citrix Solution Provider and Citrix track and resolve the issue.

Troubleshooting of installation issues is covered in the XenServer Installation Guide. Troubleshooting of Virtual Machine issues is covered in the XenServer Virtual Machine Installation Guide.

**Important**

We recommend that you follow the troubleshooting information in this chapter solely under the guidance of your Citrix Solution Provider or Citrix Support.

Citrix provides two forms of support: you can receive free self-help support via the Support site, or you may purchase our Support Services and directly submit requests by filing an online Support Case. Our free web-based resources include product documentation, a Knowledge Base, and discussion forums.

**XenServer host logs**

The XenCenter can be used to gather XenServer host information. Click on Get Server Status Report... in the Tools menu to open the Server Status Report wizard. You can select from a list of different types of information (various logs, crash dumps, etc.). The information is compiled and downloaded to the machine that XenCenter is running on. For details, see the XenCenter Help.

Additionally, the XenServer host has several CLI commands to make it simple to collate the output of logs and various other bits of system information using the utility xen-bugtool. Use the xe command host-bugreport-upload to collect the appropriate log files and system information and upload them to the Citrix Support ftp site. Please refer to the section called “host-bugreport-upload” for a full description of this command and its optional parameters. If you are requested to send a crashdump to Citrix Support, use the xe command host-crashdump-upload. Please refer to the section called “host-crashdump-upload” for a full description of this command and its optional parameters.

**Caution**

It is possible that sensitive information might be written into the XenServer host logs.

By default, the server logs report only errors and warnings. If you need to see more detailed information, you can enable more verbose logging. To do so, use the host-loglevel-set command:

```
host-loglevel-set log-level=level
```

where `level` can be 0, 1, 2, 3, or 4, where 0 is the most verbose and 4 is the least verbose.

Log files greater than 5 MB are rotated, keeping 4 revisions. The logrotate command is run hourly.
Sending log messages to a central server

Rather than have logs written to the control domain filesystem, you can configure a XenServer host to write them to a remote server. The remote server must have the syslogd daemon running on it to receive the logs and aggregate them correctly. The syslogd daemon is a standard part of all flavors of Linux and Unix, and third-party versions are available for Windows and other operating systems.

To write logs to a remote server

1. Set the syslog_destination parameter to the hostname or IP address of the remote server where you want the logs to be written:

   ```
   xe host-param-set uuid=<xenserver_host_uuid> logging:syslog_destination=<hostname>
   ```

2. Issue the command

   ```
   xe host-syslog-reconfigure uuid=<xenserver_host_uuid>
   ```

   to enforce to change. (You can also execute this command remotely by specifying the host parameter.)

Troubleshooting connections between XenCenter and the XenServer host

If you have trouble connecting to the XenServer host with XenCenter, check the following:

- Is your XenCenter an older version than the XenServer host you are attempting to connect to?
  
  The XenCenter application is backward-compatible and can communicate properly with older XenServer hosts, but an older XenCenter cannot communicate properly with newer XenServer hosts.

  To correct this issue, install a XenCenter version that is the same, or newer, than the XenServer host version.

- Is your license current?
  
  You can see the expiration date for your License Key in the XenServer host Overview tab under the Licenses section.

  Also, if you upgraded your software from version 3.2.0 to the current version, you should also have received and applied a new License file.

  For details on licensing a host, see the chapter "XenServer Licensing" in the XenServer Installation Guide.

- The XenServer host talks to XenCenter via HTTPS over port 443 (a two-way connection for commands and responses using the XenAPI), and 5900 for graphical VNC connections with paravirtual Linux VMs. If you have a firewall enabled between the XenServer host and the machine running the client software, make sure that it allows traffic from these ports.
Appendix A. CPU Allocation Guidelines

Consolidation of virtualized application workloads is most successful when all workloads are not competing for the identical physical resource (CPU, disk, network) at the same time. Workload optimization is best informed by measurements (e.g. PlateSpin PowerRecon) of the historical resource consumption of each workload on physical hardware.

This section provides a simple guideline for distributing physical CPU cores among multiple XenServer virtual machines. There are situations where this simple rule would not be appropriate, but for many use cases, two principles apply:

**Don't put more work on a server than it can do.**
Generally server time is less expensive than human time. If you put too much work on a physical server, so that VMs have to wait for CPU time, the performance will suffer, and people will be waiting for the server instead of the server waiting for people.

**Provision just enough Virtual CPUs (VCPUs) for each VM's workload.**

1. Don't give a VM more VCPUs than it needs to do its work. If the server in a VM only needs around 300 MHz, it only needs 1 VCPU. Only give it one. If it regularly has 3000MHz worth of work on your 2.5GHz quad-core box, give it two VCPUs; but don't give it three or four.
2. The number of VCPUs a VM has should reflect the amount of work it has to do. If your VM has 2 VCPUs, it should have more work than 1 physical core can do in a timely manner. If it has 4 VCPUs, it should have more work than 3 physical cores can do in a timely manner.

As a formal application of these principles for workloads that require multi-VCPU virtual machines, allocate VCPUs to maintain this constraint: 

\[(V - N) <= (P - 1)\]

where:

- \(V\) = Total VCPUs across all VMs
- \(N\) = Number of running VMs
- \(P\) = Number of Physical CPU cores
Index

A
AMD-V (AMD hardware virtualization), 3

C
CD commands, xe CLI, 71
CLI (see command line interface)
Command line interface (CLI)
  basic xe syntax, 66
  Bonding commands, 71
  CD commands, 71
  command types, 68
  console commands, 73
  event commands, 74
  host (XenServer host) commands, 75
  log commands, 83
  low-level list commands, 70
  low-level parameter commands, 69
  network commands, 84
  overview,
  parameter types, 69
  patch commands, 85
  PBD commands, 86
  PIF commands, 87
  Resource pool commands, 90
  shorthand xe syntax, 67
  special characters and syntax, 67
  Storage Manager commands, 93
  Storage repository (SR) commands, 94
  task commands, 96
  Template commands, 97
  update commands, 103
  user commands, 103
  VBD commands, 103
  VDI commands, 105
  VIF commands, 109
  VLAN commands, 111
  VM commands, 111
  xe command reference,
Console commands, xe CLI, 73
Constraints on XenServer hosts joining resource pool, 3
Creating a resource pool, 4

E
Event commands, xe CLI, 74

F
Fibre Channel storage area network (SAN), 26
Filer, NetApp, 27
FlexVol, NetApp, 27

H
Hardware virtualization
  AMD-V, 3
  Intel VT, 3
HBA (see Host bus adapter)
Host (XenServer host) commands, xe CLI, 75
Host bus adapter, 26

I
Intel VT (Intel hardware virtualization), 3
iSCSI, 25

L
Log commands, xe CLI, 83
Logical Volume Management (LVM), 24, 24
Logs, XenServer host, 124

M
Machine failures in a resource pool, 18

N
NAS (see Network attached storage (NFS))
NetApp Filer, 27
Network attached storage (NFS), 24
Network bonding commands, xe CLI, 71
Network commands, xe CLI, 84
Networking VMs,
Networking XenServer hosts
  Initial configuration after installation, 48

P
Patch commands, xe CLI, 85
PBD commands, xe CLI, 86
PIF commands, xe CLI, 87
Pool commands, xe CLI, 90

Q
QoS settings (Enterprise Edition only)
  virtual disk, 44

R
Removing XenServer host from a resource pool, 6
Requirements, for creating resource pools, 3
Resource pool, 1,
  constraints on XenServer hosts joining, 3
  coping with machine failures, 18
  creating, 4
  master, 3, 18, 18
  member, 3, 18, 18
  removing XenServer host from, 6
  requirements for creating, 3

S
SAN, 26
SR (Storage repository) commands, 94

V
Virtual machine (VM), 111

W
Xen (software) virtualization, 3
S
SAN (see Storage Area Network)
Shared network attached storage (NFS), 24
Shared storage, 5
Storage Area Network, 25
Storage Manager commands, xe CLI, 93
Storage repository (SR)
  CD-ROM, 24
  commands, xe CLI, 94
  DVD-ROM, 24
  Fibre Channel storage area network (SAN), 26
  local disk, 24
  local hotplug devices, 24
  NetApp Filer, 27
  overview, 21
  shared iSCSI storage area network (SAN), 25
  shared network attached storage (NFS), 24
  USB read/write device, 24
T
Task commands, xe CLI, 96
Template commands, xe CLI, 97
Troubleshooting
  xen-bugtool, ,
  XenServer host logs, 124, 124
U
Update commands, xe CLI, 103
User commands, xe CLI, 103
V
VBD commands, xe CLI, 103
VDI commands, xe CLI, 105
VIF (virtual interface), 46
VIF commands, xe CLI, 109
Virtual disk QoS settings (Enterprise Edition only), 44
Virtual network, 46
VLAN commands, xe CLI, 111
VM
  commands, xe CLI, 111
  networking
  Virtual disk QoS settings (Enterprise Edition only), 44
VT (Intel hardware virtualization), 3
X
xe command reference,
  basic xe syntax, 66
  Bonding commands, 71
  CD commands, 71
  command types, 68
  console commands, 73
  event commands, 74
  host (XenServer host) commands, 75
  log commands, 83
  low-level list commands, 70
  low-level parameter commands, 69
  network commands, 84
  parameter types, 69
  patch commands, 85
  PBD commands, 86
  PIF commands, 87
  Resource pool commands, 90
  shorthand xe syntax, 67
  special characters and syntax, 67
  Storage Manager commands, 93
  Storage repository (SR) commands, 94
  task commands, 96
  Template commands, 97
  update commands, 103
  user commands, 103
  VBD commands, 103
  VDI commands, 105
  VIF commands, 109
  VLAN commands, 111
  VM commands, 111
  xe commands, command line interface (CLI),
  xen-bugtool, , 124
  XenServer host
    constraints, on joining resource pool, 3
    joining in resource pools,
    logs, 124
    networks,
    requirements, for joining resource pools, 3
  Resource pools, 1
  XenServer host troubleshooting
    logs, 124
    xen-bugtool, , 124