

XenServer Administrator's Guide

4.1.0

Published March 2008

1.0 Edition

CITRIX®

XenServer Administrator's Guide: Release 4.1.0

Published March 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.:

1. United States Non-Provisional Utility Patent Application Serial Number 11/487,945, filed on July 17, 2006, and entitled “Using Writeable Page Tables for Memory Address Translation in a Hypervisor Environment”.

2. United States Non-Provisional Utility Patent Application Serial Number 11/879,338, filed on July 17, 2007, and entitled “Tracking Current Time on Multiprocessor Hosts and Virtual Machines”.

Table of Contents

1. Overview 1
    1.1. XenServer Hosts and resource pools 1
    1.2. Networking 1
    1.3. Storage 1
    1.4. Command Line Interface 2
    1.5. How this Guide relates to other documentation 2
2. XenServer Hosts and resource pools 3
    2.1. Requirements for creating resource pools 3
    2.2. Creating a resource pool 4
    2.3. Adding shared storage 4
    2.4. Installing and managing VMs on shared storage 5
    2.5. Removing a XenServer Host from a resource pool 5
    2.6. Backups 6
    2.7. Coping with machine failures 7
        2.7.1. Member failures 7
        2.7.2. Master failures 8
        2.7.3. Pool failures 8
        2.7.4. Coping with Failure due to Configuration Errors 8
        2.7.5. Physical Machine failure 9
3. Storage 11
    3.1. Overview 11
        3.1.1. Storage repositories (SRs) 11
        3.1.2. Virtual Disk Images (VDI) 11
        3.1.3. Managing Storage 12
    3.2. Storage Repository Types 12
        3.2.1. Local Disks 13
        3.2.2. Local hotplug devices 13
        3.2.3. Shared Network Attached Storage using NFS 13
        3.2.4. Shared iSCSI Storage 14
        3.2.5. Shared LVM storage over FC or iSCSI hardware HBAs 15
        3.2.6. Shared NetApp Storage 16
    3.3. Storage configuration examples 18
        3.3.1. Creating Storage Repositories 18
        3.3.2. Probing an SR 19
        3.3.3. Creating a local LVM SR (lvm) 21
        3.3.4. Creating a local EXT3 SR (ext) 21
        3.3.5. Creating a shared NFS SR (nfs) 21
        3.3.6. Creating a shared LVM over iSCSI SR using the software iSCSI initiator (lvmoiscsi) 22
        3.3.7. Creating a shared LVM over Fibre Channel or iSCSI HBA SR (lvmohba) 22
        3.3.8. Creating a shared NetApp SR over iSCSI 25
    3.4. Managing Storage Repositories 25
        3.4.1. Destroying or forgetting a SR 25
        3.4.2. Introducing a SR 26
        3.4.3. Converting local Fibre Channel SRs to shared SRs 26
        3.4.4. Moving Virtual Disk Images (VDIs) between SRs 27
        3.4.5. Managing VDIs in a Netapp SR 28
        3.4.6. Taking VDI snapshots with a Netapp SR 28
        3.4.7. Adjusting the disk IO scheduler for an LVM-based SR 28
    3.5. Managing Host Bus Adapters (HBAs) 28

3.5.1. Sample QLogic iSCSI HBA setup 29
3.5.2. Removing HBA-based FC or iSCSI device entries 29
3.6. Virtual disk QoS settings (Enterprise Edition only) 29
4. Networking 31
4.1. Networking configuration performed during Host installation 31
4.2. Managing networking configuration for standalone XenServer Hosts 32
4.2.1. Creating networks 32
4.2.2. Connecting Virtual Machines (VMs) to networks 33
4.2.3. Creating VLANs 33
4.2.4. Creating NIC bonds 34
4.2.5. Controlling Quality of Service (QoS) 36
4.2.6. Dedicating a NIC to storage traffic 36
4.2.7. Changing networking configuration options 37
4.3. Managing networking configuration for XenServer pools 38
4.3.1. Networks in resource pools 38
4.3.2. Creating NIC bonds in resource pools 38
4.3.3. Changing XenServer Host IP address configuration in resource pools..... 42
4.3.4. NIC/PIF ordering 43
4.4. Networking Troubleshooting 45
4.4.1. Diagnosing network corruption 45
4.4.2. Recovering from a bad network configuration 46
5. Command line interface 47
5.1. Basic xe syntax 47
5.2. Special characters and syntax 48
5.3. Command types 49
5.3.1. Parameter types 50
5.3.2. Low-level param commands 50
5.3.3. Low-level list commands 51
5.4. xe command reference 52
5.4.1. Bonding commands 52
5.4.2. CD commands 53
5.4.3. Console commands 54
5.4.4. Event commands 54
5.4.5. Host (XenServer Host) commands 55
5.4.6. Log commands 63
5.4.7. Network commands 64
5.4.8. Patch (update) commands 65
5.4.9. PBD commands 66
5.4.10. PIF commands 67
5.4.11. Pool commands 69
5.4.12. Storage Manager commands 71
5.4.13. SR commands 72
5.4.14. Task commands 74
5.4.15. Template commands 75
5.4.16. Update commands 80
5.4.17. User commands 80
5.4.18. VBD commands 80
5.4.19. VDI commands 83
5.4.20. VIF commands 85
5.4.21. VLAN commands 87
5.4.22. VM commands 87
6. Troubleshooting 99
6.1. XenServer Host logs 99
6.1.1. Sending log messages to a central server 99

6.2. Troubleshooting connections between XenCenter and the XenServer Host 100
6.3. Special debug boot options 100
Index 101

Chapter 1. Overview

This document is an 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
- XenServer command line interface

1.1. 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

1.2. 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

1.3. 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

1.4. 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)

1.5. 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. 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.

2.1. 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

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

- it has a static IP address (either manually assigned or via DHCP);
- it is not a member of an existing resource pool
- it has a clock synchronized to the pool master server (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
- the Linux Pack is either installed on all hosts in the pool, or not installed at all

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.

Note

The requirement for a XenServer Host to have a static IP address to be part of a resource pool also applies to the servers providing the shared NFS or iSCSI storage.

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 **xe vm-copy** CLI command or XenCenter.

2.2. Creating a resource pool

Resource pools can be created using either the XenCenter management console or the CLI.

Procedure 2.1. To join XenServer Hosts *a* and *b* into a resource pool via the CLI

1. Open a console on XenServer Host *b*.
2. Command XenServer Host *b* to join the pool on XenServer Host *a* by issuing the command:

```
xe pool-join master-address=a master-username=root master-password=password
```

The *master-address* must be set to the fully-qualified domain name of XenServer Host *a* and the *password* must be the administrator password set when XenServer Host *a* was installed.

2.3. 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.

Procedure 2.2. 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

```
xe pool-list
```

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.

2.4. Installing and managing VMs on shared storage

Procedure 2.3. Installing a Debian Etch (4.0) VM

1. Open a text console on any XenServer Host in the pool.
2. Create the Debian VM by issuing he command

```
xe vm-install template="Debian Etch 4.0" new-name-label=etch
```

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

3. 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 **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.

4. Use XenMotion to move the Debian VM to XenServer Host *b* with the command

```
xe vm-migrate vm=etch host=b --live
```

When this command returns, the VM will have been relocated to XenServer Host *b*. During this process the VM keeps running to minimize the amount of downtime.

2.5. 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.

Procedure 2.4. To remove XenServer Host *b* from a resource pool via the CLI

1. Open a text console on any XenServer Host in the pool.
2. Find the UUID corresponding to XenServer Host *b* using the command

```
xe host-list
```

3. Command XenServer Host *b* to leave the pool with the command

```
xe pool-eject host-uuid=uuid
```

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

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, or use the **xe vm-copy** CLI command.

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.

2.6. 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.

Procedure 2.5. 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.

Procedure 2.6. 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 may have to reboot to the original install CD.
- This data can only be restored to the original machine.

Procedure 2.7. 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 updata.

Warning

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

Procedure 2.8. To backup VM metadata only

- Run the command:

```
xe vm-export vm=vm-uuid filename=backup --metadata
```

2.7. 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 2.6, “Backups”.

2.7.1. Member failures

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 member (e.g. by physically rebooting it). When the connection to the member is restored, the master will mark the member as alive again.
- 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 as well.

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 forcibly halt the VMs. See Section 5.4.22.21, “vm-reset-powerstate” for more details.

2.7.2. 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 reestablish communication with its members, they 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.

2.7.3. Pool failures

In the unfortunate event that your entire resource pool fails, you will need to re-create the pool database from scratch. Be sure to regularly back up your pool-metadata using the **xe pool-dump-database** CLI command (see Section 5.4.11.2, “pool-dump-database”).

Procedure 2.9. 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 Section 5.4.11.8, “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.

2.7.4. Coping with Failure due to Configuration Errors

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

Procedure 2.10. To restore host sotfware 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".

2.7.5. 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-uuid --force
```

VMs can then be restarted using XenCenter or the CLI.

Procedure 2.11. If the master has failed, but members are running

1. Run the commands:

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

2. If the commands succeed, restart the VMs.

Procedure 2.12. If the master and all members have 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 (e.g. 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 Section 5.4.9, “PBD commands” for documentation of these commands.

3. If you have created a new storage configuration, use **pbd-plug** or XenCenter's Storage ... Repair Storage Repository menu item to use the new configuration.

4. Restart all VMs.

Procedure 2.13. 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 Host Enterprise Edition are described.

3.1. Overview

3.1.1. 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 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 or Fibre Channel SR. Finally, all hosts in a single resource pool must have at least one shared SR in common.

3.1.2. Virtual Disk Images (VDI)

There are three basic VDI types, VHD, Logical Volume Manager (LVM) and Netapp managed LUNs. Both VHD and LVM types can exist on local dedicated storage or remote shared storage. Netapp storage types must reside on a Network Appliance filer running version 7.0 or greater of Ontap.

- 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 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 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.

  VHD 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.

  Since VHD-based images involve extra metadata to provide 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.

- 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 filer 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.

3.1.3. 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 Section 5.4.13, “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 Section 5.4.9, “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 Section 5.4.19, “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 Section 5.4.18, “VBD commands”.

3.2. 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.

SR type	Description	Shared?	Sparse?	VDI Resize?	Fast Clone?
ext	VHD on Local Disk	no	yes	no	yes
nfs	VHD on Network File System	yes	yes	no	yes
lvm	Logical Volume Management on Local Disk	no	no	yes	no

SR type	Description	Shared?	Sparse?	VDI Resize?	Fast Clone?
lvmohba	Logical Volume Management over Fibre Channel or iSCSI Host Bus Adapter (HBA)	yes	no	yes	no
lvmoiscsi	Logical Volume Management over iSCSI using software initiator	yes	no	yes	no
netapp	NetApp filer using Ontap	yes	yes	yes	yes

All storage repositories in XenServer are implemented as Python scripts and stored within the control domain's file system in `/opt/xensource/sm`. Advanced users may examine and even modify these scripts to adapt storage operations to their needs. This is considered an advanced operation, and is not supported. However, visibility and customization of low-level storage management is of considerable value to some power users.

New SR implementations 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 Section 5.4.12, “Storage Manager commands”).

Creating a new SR requires the use of the sr-create command. This command creates a new SR record in the database and a corresponding PBD record. 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 for every XenServer Host in the resource pool and plugged on each XenServer Host.

3.2.1. 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.

3.2.2. 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.

3.2.3. 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 Host 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 involve extra metadata to provide 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, there are performance and access control issues to consider. 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.

Warning

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

3.2.4. 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 Section 3.2.5, “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.

3.2.4.1. 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.

3.2.5. 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 Section 3.3.1, “Creating Storage Repositories” for details on creating shared HBA-based FC and iSCSI SRs.

Note

XenServer Host support for Fibre Channel has the following restrictions:

- Dual Multipath is not supported
- Direct mapping of a LUN to a VM is not supported. 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.

3.2.6. 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 Section 3.3.8, “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 to 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.

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 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 gaurantee, 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 garauntees, 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.

A-SIS deduplication

A-SIS (Advanced Single Instance Storage) 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. A-SIS can be enabled on "thin provisioned" Netapp-based SRs and will operate according to the default filer A-SIS parameters, typically every 24 hours. It must be enabled at the point the SR is created and any custom A-SIS 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 priviledges.

Licenses

You need to have CIFS, NFS, FCP and iSCSI licenses on the NetApp filer.

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

3.3. 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.

3.3.1. 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 Section 3.4.1, “Destroying or forgetting a SR ” for details.

3.3.2. 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 server will be returned:

```
# xe sr-probe type=lvmoiscsi 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=lvmoiscsi 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>
            149455400000000000000000002000000b70200000f000000
        </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=lvmoiscsi device-config:target=192.168.1.10  \
device-config:targetIQN=192.168.1.10:filer1 \
device-config:SCSIid=149455400000000000000000002000000b70200000f000000

<?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:

SR type	device-config parameter, in order of dependency	Can be probed?	Required for sr-create?
lvmoiscsi	target	No	Yes
	chapuser	No	No
	chappassword	No	No
	targetIQN	Yes	Yes
	SCSIid	Yes	Yes
lvmohba	device	Yes	Yes
netapp	target	No	Yes

SR type	device-config parameter, in order of dependency	Can be probed?	Required for sr-create?
	username	No	Yes
	password	No	Yes
	chapuser	No	No
	chappassword	No	No
	aggregate	No	Yes
	FlexVols	No	No
	allocation	No	No
	asis	No	No
nfs	server	No	Yes
	serverpath	Yes	Yes
lvm	device	No	Yes
ext	device	No	Yes

3.3.3. Creating a local LVM SR (lvm)

Device-config parameters for lvm SRs are:

Parameter Name	Description	Required
Device	device name on the local host to use for the SR	Yes

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
```

3.3.4. Creating a local EXT3 SR (ext)

Device-config parameters for ext SRs:

Parameter Name	Description	Required
Device	device name on the local host to use for the SR	Yes

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
```

3.3.5. Creating a shared NFS SR (nfs)

Device-config parameters for nfs SRs:

Parameter Name	Description	Required
server	IP address or hostname of the NFS server	Yes
serverpath	path, including the NFS mount point, on the NFS server in which the SR will reside	Yes

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

```
xe sr-create host-uuid=<VALID_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
```

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

Device-config parameters for lvmoiscsi SRs:

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

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

```
xe sr-create host-uuid=<valid_UUID> content-type=user \
name-label="Example shared LVM over iSCSI SR" shared=true \
device-config:target=<valid_target_IP> device-config:targetIQN=<valid_target_IQN> \
device-config:SCSIid=<valid_SCSIID> \
type=lvmoiscsi
```

3.3.7. 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:

Parameter name	Description	Required
Device	HBA LUN global device path (e.g. /dev/disk/by-id/scsi-<SCSI ID>	Yes

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`

   Section 3.3.13 provides an example for configuring the QLogic iSCSI HBA. 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-360a98000686669496734463876653 36f
        </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:device=<valid_global_device_path> 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.

3.3.8. Creating a shared NetApp SR over iSCSI

Device-config parameters for netapp SRs:

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

To create a netapp SR use the following command.

```
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
```

3.4. Managing Storage Repositories

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

3.4.1. 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>
```

3.4.2. Introducing a SR

Introducing an SR that has been forgotten requires introducing an SR, creating a PBD, and manually plugging the PBD to the appropriate XenServer Host 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=149455400000000000000000002000000b70200000f000000
```

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=149455400000000000000000002000000b70200000f000000
```

4. Plug the PBD to attach the SR:

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

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

```
xe 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.

3.4.3. Converting local Fibre Channel SRs to shared SRs

Previous XenServer releases 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 4.1.
2. Ensure all hosts in the pool have the SR's LUN zoned appropriately. See Section 3.3.2, “Probing an SR ” for details on using sr-probe to verify the LUN is present on each host.
3. Convert the SR to shared:

```
xe 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.

3.4.4. 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.

3.4.4.1. 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.

3.4.4.2. 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 it's `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 vbd-destroy CLI command.
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.

3.4.5. 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 organised. 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>
```

3.4.6. Taking VDI snapshots with a Netapp SR

As outlined earlier in Section 3.2.6, “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>
```

3.4.7. 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 (Section 3.6, “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>
```

3.5. Managing Host Bus Adapters (HBAs)

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

3.5.1. 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 Section 3.3.2, “Probing an SR ” and Section 3.3.7, “Creating a shared LVM over Fibre Channel or iSCSI HBA SR (lvmohba) ” for more details.

3.5.2. 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 Section 3.4.1, “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 Section 3.3.2, “Probing an SR ” for details.

4. Remove the device entries with the following command:

```
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.

3.6. 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 Section 3.4.7, “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:

```
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 4 physical NICs per XenServer Host and up to 7 virtual network interfaces (VIFs) per VM.

XenServer 4.1 provides automated configuration and management of NICs via the xe command line interface (CLI). As VIFs, PIFs, and networks are created and configured the XenServer Host will perform all required configuration of the physical NICs.

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, and additional information that applies when using pools.

The xe CLI can be used to create, destroy and modify three types of server-side objects which represent networking entities. These objects are:

- A *PIF*, which represents a physical network interface on a XenServer Host.
- A *VIF*, which represents a virtual interface on a Virtual Machine.
- 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 the network.

Additional CLI commands 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.

The examples in this chapter assume familiarity with the xe CLI. For more information on using the xe CLI, see Chapter 5, *Command line interface*.

4.1. Networking configuration performed during Host installation

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.

4.2. Managing networking configuration for standalone XenServer Hosts

4.2.1. Creating 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.

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.

Procedure 4.1. 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.

4.2.2. Connecting Virtual Machines (VMs) to networks

When a new VM is created via the CLI, it does not contain a virtual network interface (VIF). Adding a VIF and connecting it to the desired network must be done as additional steps. When creating VMs using XenCenter, the New VM wizard contains a page allowing creation of virtual network interfaces.

Procedure 4.2. To add a new virtual interface to a VM via the CLI

1. Use **xe vm-list** to find the UUID of the VM to which you want to add the VIF. For example, if the VM is named *windows*, issue the following command to return the UUID of the VM:

   ```
   xe vm-list params=uuid name-label=windows
   ```

2. Add a VIF to the VM with the **vif-create** command, specifying the VM and network UUIDs. The UUID of the new virtual interface will be returned:

   ```
   xe vif-create vm-uuid=<VM UUID> network-uuid=<network UUID> device=0
   ```

   The device parameter is a number which uniquely identifies the virtual NIC (0, 1, 2, etc.) to a VM.

3. VIFs are automatically plugged into VMs when a VM is started, but if the VM is already running when the VIF is created, the VIF must be manually connected, or *hot-plugged*. Hot-plugging VIFs requires the XenServer Tools to be installed in the VM. Refer to the *XenServer Virtual Machine Installation Guide* for details on installing the XenServer Tools. To hot-plug a VIF to a VM use the vif-plug command:

   ```
   xe vif-plug uuid=<VIF UUID>
   ```

4.2.3. Creating 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 PIFs 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 does not include VLAN tags in packets routed to VMs.

Procedure 4.3. 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:

```
xe pif-list
```

4. Create a VLAN object specifying the desired physical PIF and VLAN tag. 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 Section 4.2.2, “Connecting Virtual Machines (VMs) to networks” for more details.

4.2.4. Creating 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/passive mode, with only one physical NIC ever in use.

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.

Also see Section 4.3.2, “Creating NIC bonds in resource pools” for details on creating NIC bonds with resource pools.

4.2.4.1. 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.

Procedure 4.4. To create a NIC bond a dual-NIC host

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 Section 4.2.4.2, “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:

```
xe pif-list device=bond0
```

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

```
xe pif-reconfigure-ip uuid=<bond PIF UUID> mode=DHCP
```

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 mgmt 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.

4.2.4.2. 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 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.

4.2.4.3. 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-ip-reconfigure** and **host-management-reconfigure** commands prior to issuing the **bond-destroy** command, otherwise connections to the host (including XenCenter) will be dropped.

4.2.5. 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:

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

4.2.6. Dedicating a NIC to storage traffic

XenServer Hosts automatically manage NICs as needed based on the related network, VIF, PIF, and bond configuration. In the default XenServer networking configuration, all network traffic to IP-based storage devices occurs over the PIF used for the management interface. To dedicate a NIC to storage traffic, it must be configured as unmanaged, and the NIC can no longer be used for XenServer networks.

Procedure 4.5. To dedicate a NIC to storage traffic

1. Use the **pif-list** command to determine which PIF corresponds to the NIC to be dedicated to storage traffic.
2. Use the **pif-forget** command to mark the NIC as unmanaged. The NIC will remain unmanaged across host restarts.
3. Perform any required IP address configuration and enable the NIC using standard Linux commands.

4. Once storage resources are available via the NIC, create the appropriate storage repositories.

To change an unmanaged NIC to managed, use the **pif-introduce** command.

Note

Use of the **pif-scan** command will reset all NICs on the host to managed.

4.2.7. Changing networking configuration options

4.2.7.1. Hostname

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

```
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.

4.2.7.2. 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:

```
pif-reconfigure-ip uuid=<PIF UUID> mode=static DNS=<new DNS IP>
```

4.2.7.3. XenServer Host IP address configuration

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 Section 5.4.10.4, “pif-reconfigure-ip” for details on the parameters of the **pif-reconfigure-ip** command.

Note

See Section 4.3.3, “Changing XenServer Host IP address configuration in resource pools” for details on changing host IP addresses in resource pools.

4.2.7.4. 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.

Procedure 4.6. 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 5, *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:

```
xe host-management-reconfigure pif-uuid=<PIF UUID>
```

4.2.7.5. 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.

4.3. Managing networking configuration for XenServer pools

4.3.1. 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 A and B are in the same pool and host A has four NICs while host B only has two, only the networks connected to PIFs corresponding to eth0 and eth1 will be valid on host B. VMs on host A with VIFs connected to networks corresponding to eth2 and eth3 will not be able to migrate to host B.

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

4.3.2. 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 is also a disruptive operation - all VMs in the pool must be shut down.

4.3.2.1. Adding NIC bonds to new resource pools

1. Install the master host and use XenCenter to create the pool.
2. Create the NIC bond on the master as follows:
    1. 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.

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

    2. Use the **host-list** command to find the UUID of the master host:

```
xe host-list
```

    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, 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:

```
xe bond-create network-uuid=<network UUID> pif-uuids=<PIF UUID 1>,<PIF UUID 2>
```

> **Note**
>
> See Section 4.2.4.2, “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:

```
xe pif-list device=<network UUID>
```

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

```
xe pif-reconfigure-ip uuid=<bond PIF UUID> mode=DHCP
```

    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 mgmt PIF UUID> mode=None
```

3. Join a member server to the pool.

   The network and bond information will be automatically replicated to the member server.

4. Move the management interface on the member server to enable the bond as follows:

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

```
xe host-list
```

   2. 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:

```
xe pif-list device=<network name> host-uuid=<host UUID>
```

   3. 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 5, *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 to be affected:

```
xe pif-reconfigure-ip uuid=<bond PIF UUID> mode=DHCP
```

   4. 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 to be affected:

```
xe host-management-reconfigure pif-uuid=<bond PIF UUID>
```

   5. 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 to be affected:

```
xe pif-reconfigure-ip uuid=<old mgmt PIF UUID> mode=None
```

5. For each member server, join it to the pool and repeat 3 and 4 to move the management interface on the member server to enable the bond.

4.3.2.2. 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.

Procedure 4.7. 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.

```
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.

```
xe vm-shutdown uuid=<VM UUID>
```

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

```
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:

```
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.

```
xe bond-create network-uuid=<network UUID> pif-uuids=<PIF UUID 1>,<PIF UUID 2>
```

Note

See Section 4.2.4.2, “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:

```
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 5, *Command line interface* for more detail on the options available for the **pif-reconfigure-ip** command. This command must be run directly on the server to be affected:

```
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 to be affected:

```
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 to be affected:

```
xe pif-reconfigure-ip uuid=<old mgmt 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.

4.3.3. Changing XenServer Host 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.

Procedure 4.8. To change 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 5, *Command line interface* for details on the parameters of the **pif-reconfigure-ip** command:

```
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:

```
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.

Procedure 4.9. 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 5, *Command line interface* for details on the parameters of the **pif-reconfigure-ip** command:

```
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:

```
xe pool-recover-slaves
```

Refer to the Section 2.7.2, “Master failures” for more information on emergency mode.

4.3.4. NIC/PIF ordering

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.

4.3.4.1. 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.

```
xe pif-list params=uuid,device,MAC,currently-attached,carrier,management, \
IP-configuration-mode

uuid ( RO)                    : 1ef8209d-5db5-cf69-3fe6-0e8d24f8f518
                  device ( RO): eth0
                     MAC ( RO): 00:19:bb:2d:7e:8a
       currently-attached ( RO): true
              management ( RO): true
    IP-configuration-mode ( RO): DHCP
                 carrier ( RO): true


uuid ( RO)                    : 829fd476-2bbb-67bb-139f-d607c09e9110
                  device ( RO): eth1
                     MAC ( RO): 00:19:bb:2d:7e:7a
       currently-attached ( RO): false
              management ( RO): false
    IP-configuration-mode ( RO): None
                 carrier ( RO): true
```

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.

4.3.4.2. 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:

   ```
   xe host-management-disable
   ```

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

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

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

```
xe pif-introduce device=eth0 host-uuid=<host UUID> mac=00:19:bb:2d:7e:7a
xe pif-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:

```
xe pif-list params=uuid,device,MAC
```

6. Use the **pif-reconfigure-ip** command to reset the management interface IP addressing configuration. See Chapter 5, *Command line interface* for details on the parameters of the **pif-reconfigure-ip** command.

```
xe pif-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:

```
xe host-management-reconfigure pif-uuid=728d9e7f-62ed-a477-2c71-3974d75972eb
```

4.4. 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.

4.4.1. 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:

```
$ xe pif-list device=eth0
```

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

```
$ xe pif-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.

4.4.2. 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:

- It is recommended 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 configurations for a minimal network configuration (including one interface and one bridge; for example, eth0 on the xenbr0 bridge).

Chapter 5. 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 `C:\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-4.1.0-7843p.i386.rpm` on the Linux Pack CD, as follows:

rpm -ivh xe-cli-4.1.0-7843p.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

5.1. 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

Shorthand syntax is also available for remote connection arguments:

-u	username
-pw	password
-pwf	password file
-p	port
-s	server

Example: On a remote XenServer Host:

xe 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.

5.2. 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

```
xe vm-l
```

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 Section 5.1, “Basic xe syntax” for details.

5.3. 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
- vlan
- vm

Note that not every value of *class* has the full set of *class*-**param-** commands; some have just a subset.

5.3.1. 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.

5.3.2. 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.

5.3.3. 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
```

5.4. xe command reference

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

5.4.1. Bonding commands

Commands for working with network bonds, for resilience with physical interface failover. See Section 4.2.4, “Creating NIC bonds” 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:

Parameter Name	Description	Type
uuid	unique identifier/object reference for the bond	read only
master	UUID for the master bond PIF	read only
members	set of UUIDs for the underlying bonded PIFs	read only set parameter

5.4.1.1. 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.

5.4.1.2. bond-destroy

`host-bond-destroy` uuid=*`bond UUID`*

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

5.4.2. CD commands

Commands for working with physical CD/DVD drives on XenServer Hosts.

CD parameters

CDs have the following parameters:

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

Parameter Name	Description	Type
managed	true if the device is managed	read only
xenstore-data	Data to be inserted into the xenstore tree	read only map parameter
sm-config	names and descriptions of storage manager device config keys	read only map parameter

5.4.2.1. 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.

5.4.3. 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 Section 5.3.2, “Low-level param commands” for details.

Console parameters

Consoles have the following parameters:

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

5.4.4. Event commands

Commands for working with events.

Event classes

Event classes are listed in the following table:

Class name	Description
pool	A pool of physical hosts
vm	A Virtual Machine
host	A physical host
network	A virtual network
vif	A virtual network interface
pif	A physical network interface (note separate VLANs are represented as several PIFs)
sr	A storage repository
vdi	A virtual disk image
vbd	A virtual block device
pbd	The physical block devices through which hosts access SRs

5.4.4.1. 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.

5.4.5. Host (XenServer Host) commands

Commands for interacting with XenServer Hosts.

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 Section 5.3.2, “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:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the XenServer Host	read only
name-label	The name of the XenServer Host	read/write
name-description	The description string of the XenServer Host	read only
enabled	false if disabled which prevents any new VMs from starting on them, which prepares the XenServer Hosts to be shut down or rebooted; true if the host is currently enabled	read only
API-version-major	major version number	read only
API-version-minor	minor version number	read only
API-version-vendor		read only
API-version-vendor-implementation	details of vendor implementation	read only map parameter
logging	logging configuration	read/write map parameter
suspend-image-sr-uuid	the unique identifier/object reference for the SR where suspended images are put	read/write
crash-dump-sr-uuid	the unique identifier/object reference for the SR where crash dumps are put	read/write
software-version	list of versioning parameters and their values	read only map parameter
capabilities	list of Xen versions that the XenServer Host can run	read only set parameter
other-config	A list of key/value pairs that specify additional configuration parameters for the XenServer Host	read/write map parameter
hostname	XenServer Host hostname	read only
address	XenServer Host IP address	read only
supported-bootloaders	list of bootloaders that the XenServer Host supports, for example, pygrub, eliloader	read only set parameter

Parameter Name	Description	Type
memory-total	total amount of physical RAM on the XenServer Host, in bytes	read only
memory-free	total amount of physical RAM remaining that can be allocated to VMs, in bytes	read only
host-metrics-last-updated	Timestamp of the date and time that the metrics for a XenServer Host were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
host-metrics-live	true if the host is operational	read only
logging	The `syslog_destination` key can be set to the hostname of a remote listening syslog service.	read/write map parameter
allowed-operations	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.	read only set parameter
current-operations	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.	read only set parameter
patches	Set of host patches	read only set parameter

XenServer Hosts contain some other objects that also have parameter lists.

CPUs on XenServer Hosts have the following parameters:

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

Crash dumps on XenServer Hosts have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the crashdump	read only
host	XenServer Host the crashdump corresponds to	read only
timestamp	Timestamp of the date and time that the crashdump occurred, in the form *yyyymmdd-hhmmss-ABC*, where *ABC* is the timezone indicator, for example, GMT	read only
size	size of the crashdump, in bytes	read only

5.4.5.1. 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.

5.4.5.2. 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.

5.4.5.3. host-crashdump-destroy

`host-crashdump-destroy` uuid=*`crashdump UUID`*

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

5.4.5.4. 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.

5.4.5.5. 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.

5.4.5.6. 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.

5.4.5.7. 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.

5.4.5.8. 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.

5.4.5.9. 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 a random other host in the pool is chosen as the pool master and control is transferred over to it.

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.

5.4.5.10. 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.

5.4.5.11. 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 Section 5.4.5.12, “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).

5.4.5.12. 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 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.

Attribute	Description
key	A unique identifier for the capability.
content-type	Can be either *text/plain* or *application/data*. Indicates whether a UI can render the entries for human consumption.
default-checked	Can be either *yes* or *no*. Indicates whether a UI should select this entry by default.
min-size, max-size	Indicates an approximate range for the size, in bytes, of this entry. *-1* indicates that the size is unimportant.
min-time, max-time	

Attribute	Description
	Indicate an approximate range for the time, in seconds, taken to collect this entry. -1 indicates the time is unimportant.
pii	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: • *no* â€“ no PII will be in these entries • *yes* â€“ PII will likely or certainly be in these entries • *maybe* â€“ you might wish to audit these entries for PII • *if_customized* â€“ 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 customizations. This is used in particular for the networking scripts in the control domain. 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).

5.4.5.13. host-is-in-emergency-mode

`host-is-in-emergency-mode`

Determines 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.

5.4.5.14. 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.

5.4.5.15. host-license-view

`host-license-view` [host-uuid=*`XenServer Host UUID`*]

Displays the contents of the XenServer Host license.

5.4.5.16. 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.

5.4.5.17. 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.

5.4.5.18. host-management-reconfigure

`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 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.

5.4.5.19. host-reboot

`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 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 you make one of the members into the master.

5.4.5.20. host-restore

`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.

5.4.5.21. 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.

5.4.5.22. 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 you make one of the members into the master.

5.4.5.23. 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.

5.4.6. Log commands

Commands for working with logs.

5.4.6.1. log-get-keys

`log-get-keys`

List the keys of all of the logging subsystems.

5.4.6.2. log-reopen

`log-reopen`

Reopen all loggers. Use this for rotating log files.

5.4.6.3. 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 or error.

5.4.7. 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 Section 5.3.2, “Low-level param commands” for details.

Network parameters

Networks have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the network	read only
name-label	The name of the network	read only
name-description	The description text of the network	read only
VIF-uuids	A list of unique identifiers of the VIFs (virtual network interfaces) that are attached from VMs to this network	read only set parameter
PIF-uuids	A list of unique identifiers of the PIFs (physical network interfaces) that are attached from XenServer Hosts to this network	read only set parameter
bridge	name of the bridge corresponding to this network on the local XenServer Host	read only

Parameter Name	Description	Type
other-config	A list of key/value pairs that specify additional configuration parameters for the network.	read/write map parameter

5.4.7.1. network-create

`network-create` name-label=*`name for network`* [name-description=*`descriptive text`*]

Creates a new network.

5.4.7.2. network-destroy

`network-destroy` uuid=*`network UUID`*

Destroys an existing network.

5.4.8. 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 Section 5.4.16, “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 Section 5.3.2, “Low-level param commands” for details.

Patch parameters

Patches have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the patch	read only
host-uuid	The unique identifier for the XenServer Host host to query	read only
name-label	The name of the patch	read only
name-description	The description string of the patch	read only
applied	Whether or not the patch has been applied; true or false	read only
size	Whether or not the patch has been applied; true or false	read only

5.4.8.1. patch-apply

`patch-apply` uuid=*`patch file UUID`*

Apply the specified patch file.

5.4.8.2. patch-clean

`patch-clean` uuid=*`patch file UUID`*

Delete the specified patch file from the XenServer Host.

5.4.8.3. patch-pool-apply

`patch-pool-apply` uuid=*`patch UUID`*

Apply the specified patch to all XenServer Hosts in the pool.

5.4.8.4. patch-precheck

`patch-precheck` uuid=*`patch UUID`* host-uuid=*`host UUID`*

Run the prechecks contained within the specified patch on the specified XenServer Host.

5.4.8.5. 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.

5.4.9. 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 Section 5.3.2, “Low-level param commands” for details.

PBD parameters

PBDs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the PBD.	read only
sr-uuid	the storage repository that the PBD points to	read only
device-config	additional configuration information that is provided to the XenServer Host's SR-backend-driver	read only map parameter
currently-attached	Is the SR currently attached on this host? True or False	read only
host-uuid	UUID of the physical machine on which the PBD is available	read only

5.4.9.1. pbd-create

`pbd-create` host-uuid=*`UUID of XenServer 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 Chapter 3, *Storage*.

5.4.9.2. pbd-destroy

`pbd-destroy` uuid=*UUID of PBD*

Destroy the specified PBD.

5.4.9.3. 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.

5.4.9.4. pbd-unplug

`pbd-unplug` uuid=*UUID of PBD*

Attempts to unplug the PBD from the XenServer Host.

5.4.10. 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 Section 5.3.2, “Low-level param commands” for details.

PIF parameters

PIFs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the PIF	read only
device	machine-readable name of the interface (for example, eth0)	read only
MAC	the MAC address of the PIF	read only
physical	If true, the PIF points to an actual physical network interface	read only
currently-attached	Is the PIF currently attached on this host? true or false	read only
MTU	Maximum Transmission Unit of the PIF in bytes.	read only

Parameter Name	Description	Type
VLAN	VLAN tag for all traffic passing through this interface; -1 indicates no VLAN tag is assigned	read only
bond-master-of	The UUID of the bond this PIF is the master of (if any)	read only
bond-slave-of	The UUID of the bond this PIF is the slave of (if any)	read only
management	Is this PIF designated to be a management interface for the control domain	read only
network-uuid	the unique identifier/object reference of the virtual network to which this PIF is connected	read only
network-name-label	the name of the of the virtual network to which this PIF is connected	read only
host-uuid	the unique identifier/object reference of the XenServer Host to which this PIF is connected	read only
host-name-label	the name of the XenServer Host to which this PIF is connected	read only
IP-configuration-mode	Type of network address configuration used; DHCP or static	read only
IP	IP address of the PIF, defined here if IP-configuration-mode is static; undefined if DHCP	read only
netmask	Netmask of the PIF, defined here if IP-configuration-mode is static; undefined if supplied by DHCP	read only
gateway	Gateway address of the PIF, defined here if IP-configuration-mode is static; undefined if supplied by DHCP	read only
DNS	DNS address of the PIF, defined here if IP-configuration-mode is static; undefined if supplied by DHCP	read only
io_read_kbs	average read rate in kB/s for the device	read only
io_write_kbs	average write rate in kB/s for the device	read only
carrier	link state for this device	read only
vendor-id	the ID assigned to NIC's vendor	read only
vendor-name	the NIC vendor's name	read only
device-id	the ID assigned by the vendor to this NIC model	read only
device-name	the name assigned by the vendor to this NIC model	read only
speed	Data transfer rate of the NIC	read only
duplex	Duplexing mode of the NIC; full or half	read only
pci-bus-path		read only

Parameter Name	Description	Type
pif-metrics-last-updated	Timestamp of the date and time that the metrics for this PIF were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
other-config	additional configuration	read/write map parameter

5.4.10.1. pif-forget

`pif-forget` uuid=*`UUID of PIF`*

Destroy the specified PIF object on a particular host.

5.4.10.2. 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.

5.4.10.3. pif-plug

`pif-plug` uuid=*`UUID of PIF`*

Attempt to bring up the specified physical interface.

5.4.10.4. 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.

5.4.10.5. pif-scan

`pif-scan` host-uuid=*`UUID of XenServer Host`*

Scan for new physical interfaces on a XenServer Host.

5.4.10.6. pif-unplug

`pif-unplug` uuid=*`UUID of PIF`*

Attempt to bring down the specified physical interface.

5.4.11. Pool commands

Commands for working with pools. A *pool* is a 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 Section 5.3.2, “Low-level param commands” for details.

Pool parameters

Pools have the following parameters:

Parameter Name	Description	Type
uuid	the unique identifier/object reference for the pool	read only
name-label	The name of the pool	read/write
name-description	The description string of the pool	read/write
master	the unique identifier/object reference of XenServer Host designated as the pool's master	read only
default-SR	the unique identifier/object reference of the default SR for the pool	read/write
crash-dump-SR	the unique identifier/object reference of the SR where any crash dumps for pool members are saved	read/write
suspend-image-SR	the unique identifier/object reference of the SR where suspended VMs on pool members are saved	read/write
other-config	A list of key/value pairs that specify additional configuration parameters for the pool	read/write map parameter
supported-sr-types	SR types that can be used by this pool	read only

5.4.11.1. 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.

5.4.11.2. 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.

5.4.11.3. pool-eject

`pool-eject` host-uuid=*`UUID of XenServer Host to eject`*

Instruct the specified XenServer Host to leave an existing pool.

5.4.11.4. pool-emergency-reset-master

`pool-emergency-reset-master` master-address=*`address of the pool's master XenServer Host`*

Instruct a member XenServer Host to reset its master address.

5.4.11.5. pool-emergency-transition-to-master

`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 Section 5.4.17, “User commands”).

5.4.11.6. pool-join

`pool-join` master-address=*`address`* master-username=*`username`* master-password=*`password`*

Instruct a XenServer Host to join an existing pool.

5.4.11.7. 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.

5.4.11.8. pool-restore-database

`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.

5.4.11.9. pool-sync-database

`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.

5.4.12. 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 Section 5.3.2, “Low-level param commands” for details.

SM parameters

SMs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the SM plugin	read only
name-label	The name of the SM plugin	read only
name-description	The description string of the SM plugin	read only
type	The SR type that this plugin connects to	read only
vendor	Name of the vendor who created this plugin	read only
copyright	copyright statement for this SM plugin	read only
required-api-version	Minimum SM API version required on the XenServer Host	read only
configuration	names and descriptions of device configuration keys	read only
capabilities	capabilities of the SM plugin	read only

5.4.13. 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 Section 5.3.2, “Low-level param commands” for details.

SR parameters

SRs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the SR	read only
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

Parameter Name	Description	Type
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 utilisation 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 SRâ€™s content. Currently used only to distinguish ISO libraries from other SRs. For storage repositories that store a library of ISOs, the *content-type* must be set to *iso*. In other cases, it is recommended that this be set either to empty, or the string 'user'.	read only
shared	true if this SR is capable of being shared between multiple XenServer Hosts; false otherwise	read/write
other-config	A list of key/value pairs that specify additional configuration parameters for the SR .	read/write map parameter
host	The storage repository host name	read only
virtual-allocation	sum of virtual_sizes of all VDIs in this storage repository (in bytes)	read only
sm-config	SM dependent data	read only map parameter

5.4.13.1. 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 Chapter 3, *Storage* for details of these parameters across the different storage backends.

5.4.13.2. sr-destroy

`sr-destroy` uuid=*SR UUID*

Destroys the specified SR on the XenServer Host.

5.4.13.3. 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).

5.4.13.4. 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.

5.4.13.5. 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 Chapter 3, *Storage* for details of these parameters across the different storage backends.

5.4.13.6. sr-scan

`sr-scan` uuid=*`SR UUID`*

Force an SR scan, syncing the xapi database with VDIs present in the underlying storage substrate.

5.4.14. 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 Section 5.3.2, “Low-level param commands” for details.

Task parameters

Tasks have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the Task	read only
name-label	The name of the Task	read only

Parameter Name	Description	Type
name-description	The description string of the Task	read only
resident-on	The unique identifier/object reference of the host on which the task is running	read only
status	current status of the Task	read only
progress	if the Task is still pending, this field contains the estimated percentage complete, from 0. to 1. If the Task has completed, successfully or unsuccessfully, this should be 1.	read only
type	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	read only
result	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	read only
error_info	if the Task has failed, this parameter contains the set of associated error strings; otherwise, this parameter's value is undefined	read only
allowed_operations	list of the operations allowed in this state	read only
created	Time the task has been created	read only
finished	Time task finished (i.e. succeeded or failed). If task-status is pending, then the value of this field has no meaning	read only

5.4.14.1. task-cancel

`task-cancel` [uuid=*`Task UUID`*]

Direct the specified Task to cancel and return.

5.4.15. 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 Section 5.3.2, “Low-level param commands” for details.

Template parameters

Templates have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the template	read only
name-label	The name of the template	read/write
name-description	The description string of the template	read/write
user-version	string for creators of VMs and templates to put version information	read/write
is-a-template	true if this is a template. Template VMs can never be started, they are used only for cloning other VMs Note that setting *is-a-template* using the CLI is not supported.	read/write
is-control-domain	true if this is a control domain (domain 0 or a driver domain)	read only
power-state	Current power state; always *halted* for a template	read only
power-state	Current power state; always *halted* for a template	read only
memory-dynamic-max	dynamic maximum memory in bytes. Currently unused, but if changed the following constraint must be obeyed: *memory_static_max* >= *memory_dynamic_max* >= *memory_dynamic_min* >= *memory_static_min*.	read/write
memory-dynamic-min	dynamic minimum memory in bytes. Currently unused, but if changed the same constraints for *memory-dynamic-max* must be obeyed.	read/write
memory-static-max	statically-set (absolute) maximum memory in bytes. This is the main value used to determine the amount of memory assigned to a VM.	read/write
memory-static-min	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 value is currently unused in normal operation, but the previous constraint must be obeyed.	read/write
suspend-VDI-uuid	The VDI that a suspend image is stored on (has no meaning for a template)	read only
VCPUs-params	configuration parameters for the selected VCPU policy.	read/write map parameter

Parameter Name	Description	Type
	You can tune a VCPU's pinning with `xe vm-param-set uuid=<VM UUID> VCPUs` 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` `xe vm-param-set uuid=<VM UUID> VCPUs` 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.	
VCPUs-max	Maximum number of VCPUs	read/write
VCPUs-at-startup	Boot number of VCPUs	read/write
actions-after-crash	action to take if a VM based on this template crashes	read/write
console-uuids	virtual console devices	read only set parameter
platform	platform-specific configuration	read/write map parameter
allowed-operations	list of the operations allowed in this state	read only set parameter
current-operations	A list of the operations that are currently in progress on this template	read only set parameter
allowed-VBD-devices	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).	read only set parameter

Parameter Name	Description	Type
allowed-VIF-devices	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).	read only set parameter
HVM-boot-policy		read/write
HVM-boot-params	The `order` key controls the HVM guest boot order, represented as a string where each character is a boot method: "*d*" for the CD/DVD, "*c*" for the root disk, and "*n*" for network PXE boot. The default is "*dc*".	read/write map parameter
PV-kernel	path to the kernel	read/write
PV-ramdisk	path to the initrd	read/write
PV-args	string of kernel command line arguments	read/write
PV-legacy-args	string of arguments to make legacy VMs based on this template boot	read/write
PV-bootloader	name of or path to bootloader	read/write
PV-bootloader-args	string of miscellaneous arguments for the bootloader	read/write
last-boot-CPU-flags	describes the CPU flags on which a VM based on this template was last booted; not populated for a template	read only
resident-on	the XenServer Host on which a VM based on this template is currently resident; appears as <not in database> for a template	read only
affinity	a XenServer Host which a VM based on this template has preference for running on; used by the **xe vm-start** command to decide where to run the VM	read/write
other-config	A list of key/value pairs that specify additional configuration parameters for the template	read/write map parameter
start-time	Timestamp of the date and time that the metrics for a VM based on this template were read, in the form *yyyymmdd*T*hh: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	read only
install-time	Timestamp of the date and time that the metrics for a VM based on this template were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT); set to 1 Jan	read only

Parameter Name	Description	Type
	1970 Z (beginning of Unix/POSIX epoch) for a template	
memory-actual	The actual memory being used by a VM based on this template; 0 for a template	read only
VCPUs-number	The number of virtual CPUs assigned to a VM based on this template; 0 for a template	read only
VCPUs-utilisation	A list of virtual CPUs and their weight	read only map parameter
vm-metrics-last-updated	Timestamp of the date and time that the metrics for a VM based on this template were read, in the form *yyyymmdd*T*hh: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	read only
os-version	the version of the operating system for a VM based on this template; appears as <not in database> for a template	read only map parameter
PV-drivers-version	the versions of the paravirtualized drivers for a VM based on this template; appears as <not in database> for a template	read only map parameter
PV-drivers-up-to-date	flag for latest version of the paravirtualized drivers for a VM based on this template; appears as <not in database> for a template	read only
memory	memory metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
disks	disk metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
networks	network metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
other	other metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
guest-metrics-last-updated	timestamp when the last write to these fields was performed by the in-guest agent, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
actions-after-shutdown	action to take after the VM has shutdown	read/write
actions-after-reboot	action to take after the VM has rebooted	read/write

Parameter Name	Description	Type
possible-hosts	list of hosts that could potentially host the VM	read only
HVM-shadow-multiplier	multiplier applied to the amount of shadow that will be made available to the guest	read/write
dom-id	domain ID (if available, -1 otherwise)	read only
recommendations	An XML specification of recommended values and ranges for properties of this VM	read only
xenstore-data	data to be inserted into the xenstore tree (/local/domain/<domid>/vm-data) after the VM is created.	read/write map parameter

5.4.15.1. 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.

5.4.16. 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 Section 5.4.8, “Patch (update) commands” for details.

5.4.16.1. 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.

5.4.17. User commands

5.4.17.1. 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.

5.4.18. 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 Section 5.3.2, “Low-level param commands” for details.

VBD parameters

VBDs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the VBD	read only
vm-uuid	The unique identifier/object reference for the VM this VBD is attached to	read only
vm-name-label	The name of the VM this VBD is attached to	read only
vdi-uuid	The unique identifier/object reference for the VDI this VBD is mapped to	read only
vdi-name-label	The name of the VDI this VBD is mapped to	read only
empty	if true, this represents an empty drive	read only
device	the device seen by the guest, for example hda1	read only
userdevice	user-friendly device name	read/write
bootable	true if this VBD is bootable	read/write
mode	the mode the VBD should be mounted with	read/write
type	how the VBD appears to the VM, for example disk or CD	read/write
currently-attached	Is the VBD currently attached on this host? true or false	read only
storage-lock	true if a storage-level lock was acquired	read only
status-code	error/success code associated with the last attach operation	read only
status-detail	error/success information associated with the last attach operation status	read only
qos_algorithm_type	the QoS algorithm to use	read/write
qos_algorithm_params	parameters for the chosen QoS algorithm	read/write map parameter
qos_supported_algorithms	supported QoS algorithms for this VBD	read only set parameter
io_read_kbs	average read rate in kB/s for this VBD	read only
io_write_kbs	average write rate in kB/s for this VBD	read only
vbd-metrics-last-updated	Timestamp of the date and time that the metrics for a VM based on this template were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
allowed-operations	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.	read only set parameter
current-operations		read only set parameter

Parameter Name	Description	Type
	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.	
unpluggable	true if this VBD will support hot-unplug	read/write
attachable	true if the device can be attached	read only
other-config	additional configuration	read/write map parameter

5.4.18.1. 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.

5.4.18.2. vbd-destroy

`vbd-destroy` uuid=*`UUID of VBD`*

Destroy the specified VBD.

5.4.18.3. 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.

5.4.18.4. 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 removeable type (a physical CD or an ISO); otherwise an error message "VBD_NOT_REMOVABLE_MEDIA" is returned.

5.4.18.5. vbd-plug

`vbd-plug` uuid=*`UUID of VBD`*

Attempt to attach the VBD while the VM is in the running state.

5.4.18.6. vbd-unplug

`vbd-unplug` uuid=*`UUID of VBD`*

Attempts to detach the VBD from the VM while it is in the running state.

5.4.19. 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 Section 5.3.2, “Low-level param commands” for details.

VDI parameters

VDIs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the VDI	read only
name-label	The name of the VDI	read/write
name-description	The description string of the VDI	read/write
allowed-operations	a list of the operations allowed in this state	read only set parameter
current-operations	a list of the operations that are currently in progress on this VDI	read only set parameter
sr-uuid	SR in which the VDI resides	read only
vbd-uuids	a list of VBDs that refer to this VDI	read only set parameter
crashdump-uuids	list of crash dumps that refer to this VDI	read only set parameter
virtual-size	size of disk as presented to the VM, in bytes. Note that, depending on storage backend type, the size may not be respected exactly	read only
physical-utilisation	amount of physical space that the VDI is currently taking up on the SR, in bytes	read only
type	type of VDI, for example, System or User	read only
sharable	true if this VDI may be shared	read only
read-only	true if this VDI can only be mounted read-only	read only
storage-lock	true if this VDI is locked at the storage level	read only
parent	References the parent VDI, if this VDI is part of a chain	read only
missing	true if SR scan operation reported this VDI as not present	read only
other-config	additional configuration information for this VDI	read/write map parameter
sr-name-label	name of the containing storage repository	read only

Parameter Name	Description	Type
location	location information	read only
managed	true if the CDI is managed	read only
xenstore-data	data to be inserted into the xenstore tree (/local/domain/0/backend/vbd/<domid>/<device-id>/sm-data) after the VDI is attached. This is generally set by the SM backends on vdi_attach.	read only map parameter
sm-config	SM dependent data	read only map parameter

5.4.19.1. 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.

5.4.19.2. 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.

5.4.19.3. 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.

5.4.19.4. 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.

5.4.19.5. 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.

5.4.19.6. vdi-import

`vdi-import` uuid=*`UUID of VDI`* filename=*`filename of raw VDI`*

Import a raw VDI.

5.4.19.7. 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 misc user-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.

5.4.19.8. vdi-resize

`vdi-resize` uuid=*`VDI UUID`* disk-size=*`new size for disk`*

Resize the VDI specified by UUID.

5.4.19.9. vdi-snapshot

`vdi-snapshot` uuid=*`UUID of the VDI`* [driver-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.

5.4.19.10. 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.

5.4.20. 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 Section 5.3.2, “Low-level param commands” for details.

VIF parameters

VIFs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the VIF	read only
vm-uuid	The unique identifier/object reference for the VM that this VIF resides on	read only
vm-name-label	The name of the VM that this VIF resides on	read only
allowed-operations	a list of the operations allowed in this state	read only set parameter
current-operations	a list of the operations that are currently in progress on this VIF	read only set parameter
device	integer label of this VIF, indicating the order in which VIF backends were created	read only
MAC	MAC address of VIF, as exposed to the VM	read only

MTU	Maximum Transmission Unit of the VIF in bytes. This parameter is read-only, but you can override the MTU setting with the *mtu* key via the other-config map parameter. For example, to reset the MTU on a virtual NIC to use jumbo frames:
	
	`xe vif-param-set uuid=<VIF UUID> \`
	`other-config:mtu=9000`
		read only

currently-attached	true if the device is currently attached	read only
qos_algorithm_type	QoS algorithm to use	read/write
qos_algorithm_params	parameters for the chosen QoS algorithm	read/write map parameter
qos_supported_algorithms	supported QoS algorithms for this VIF	read only set parameter
other-config	A list of key/value pairs that specify additional configuration parameters for this VIF.	read/write map parameter
network-uuid	The unique identifier/object reference of the virtual network to which this VIF is connected	read only
network-name-label	The descriptive name of the virtual network to which this VIF is connected	read only
io_read_kbs	average read rate in kB/s for this VIF	read only
io_write_kbs	average write rate in kB/s for this VIF	read only
vif-metrics-last-updated	Timestamp of the date and time that the metrics for the VIF were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the	read only

Parameter Name	Description	Type
	single-letter military timezone indicator, for example, Z for UTC (GMT)	

5.4.20.1. 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`*.

5.4.20.2. vif-destroy

`vif-destroy` uuid=*`UUID of VIF`*

Destroy a VIF.

5.4.20.3. vif-plug

`vif-plug` uuid=*`UUID of VIF`*

Attempt to attach the VIF while the VM is in the running state.

5.4.20.4. vif-unplug

`vif-unplug` uuid=*`UUID of VIF`*

Attempts to detach the VIF from the VM while it is in the running state.

5.4.21. 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 Section 5.4.10, “PIF commands”). For example, to list VLANs you need to use **xe pif-list**.

5.4.21.1. 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.

5.4.21.2. vlan-destroy

`vlan-destroy` uuid=*`UUID of PIF mapped to VLAN`*

Destroy a VLAN. Requires the UUID of the PIF that represents the VLAN.

5.4.22. 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 Section 5.3.2, “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.

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the VM	read only
name-label	The name of the VM	read/write
name-description	The description string of the VM	read/write
user-version	string for creators of VMs and templates to put version information	read/write
is-a-template	false unless this is a template; template VMs can never be started, they are used only for cloning other VMs Note that setting *is-a-template* using the CLI is not supported.	read/write
is-control-domain	true if this is a control domain (domain 0 or a driver domain)	read only
power-state	Current power state	read only
memory-dynamic-max	dynamic maximum in bytes	read/write
memory-dynamic-min	dynamic minimum in bytes	read/write
memory-static-max	statically-set (absolute) maximum in bytes. If you want to change this value, the VM must be shut down.	read/write
memory-static-min	statically-set (absolute) minimum in bytes If you want to change this value, the VM must be shut down.	read/write

Parameter Name	Description	Type
suspend-VDI-uuid	The VDI that a suspend image is stored on	read only

VCPUs-params	configuration parameters for the selected VCPU policy.
	
	You can tune a VCPU's pinning with
	
	`xe vm-param-set uuid=<VM UUID> VCPUs`
	
	The selected VM 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=<template UUID> VCPUs-params:weight=512`
	`xe vm-param-set uuid=<template UUID> VCPUs-params:cap=100`
	
	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.
	
	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.
		read/write map parameter

VCPUs-max	Maximum number of virtual CPUs.	read/write
VCPUs-at-startup	Boot number of virtual CPUs	read/write
actions-after-crash	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	read/write

Parameter Name	Description	Type
	destroy (no coredump and leave VM halted).	
console-uuids	virtual console devices	read only set parameter
platform	platform-specific configuration	read/write map parameter
allowed-operations	list of the operations allowed in this state	read only set parameter
current-operations	A list of the operations that are currently in progress on the VM	read only set parameter
allowed-VBD-devices	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).	read only set parameter
allowed-VIF-devices	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).	read only set parameter
HVM-boot-policy		read/write
HVM-boot-params		read/write map parameter
HVM-shadow-multiplier	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.	read/write
PV-kernel	path to the kernel	read/write
PV-ramdisk	path to the initrd	read/write
PV-args	string of kernel command line arguments	read/write
PV-legacy-args	string of arguments to make legacy VMs boot	read/write
PV-bootloader	name of or path to bootloader	read/write
PV-bootloader-args	string of miscellaneous arguments for the bootloader	read/write
last-boot-CPU-flags	describes the CPU flags on which the VM was last booted	read only
resident-on	the XenServer Host on which a VM is currently resident	read only
affinity	a XenServer Host which the VM has preference for running on; used by the **xe vm-start** command to decide where to run the VM	read/write
other-config	A list of key/value pairs that specify additional configuration parameters for the VM For example, a VM will be started automatically after host boot if the other-	read/write map parameter

Parameter Name	Description	Type
	config parameter includes the key/value pair *auto_poweron: true*	
start-time	Timestamp of the date and time that the metrics for the VM were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
install-time	Timestamp of the date and time that the metrics for the VM were read, in the form *yyyymmdd*T*hh:mm:ss z*, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
memory-actual	The actual memory being used by a VM	read only
VCPUs-number	The number of virtual CPUs assigned to the VM For a paravirtualized Linux VM, this number can differ from *VCPUS-max* and can be changed without rebooting the VM via the **vm-vcpu-hotplug** command. See Section 5.4.22.27, “vm-vcpu-hotplug”. Windows VMs always run with the number of vCPUs set to *VCPUs-max* and must be rebooted to change this value. Note that performance will drop sharply if you set *VCPUs-number* to a value greater than the number of physical CPUs on the XenServer Host.	read only
VCPUs-utilisation	A list of virtual CPUs and their weight	read only map parameter
os-version	the version of the operating system for the VM	read only map parameter
PV-drivers-version	the versions of the paravirtualized drivers for the VM	read only map parameter
PV-drivers-up-to-date	flag for latest version of the paravirtualized drivers for the VM	read only
memory	memory metrics reported by the agent on the VM	read only map parameter
disks	disk metrics reported by the agent on the VM	read only map parameter
networks	network metrics reported by the agent on the VM	read only map parameter
other	other metrics reported by the agent on the VM	read only map parameter
guest-metrics-last-updated	timestamp when the last write to these fields was performed by the in-guest agent, in the form *yyyymmdd*T*hh:mm:ss*	read only

Parameter Name	Description	Type
	z, where *z* is the single-letter military timezone indicator, for example, Z for UTC (GMT)	
actions-after-shutdown	action to take after the VM has shutdown	read/write
actions-after-reboot	action to take after the VM has rebooted	read/write
possible-hosts	potential hosts of this VM	read only
dom-id	domain ID (if available, -1 otherwise)	read only
recommendations	An XML specification of recommended values and ranges for properties of this VM	read only
xenstore-data	data to be inserted into the xenstore tree (/local/domain/<domid>/vm-data) after the VM is created	read/write map parameter
vm-metrics-last-updated	time when VM metrics were last updated	read only

5.4.22.1. 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.

5.4.22.2. vm-cd-eject

`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.

5.4.22.3. vm-cd-insert

`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.

5.4.22.4. vm-cd-list

`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.

5.4.22.5. vm-cd-remove

`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.

5.4.22.6. vm-clone

`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.

5.4.22.7. 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:

```
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.

5.4.22.8. 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.

5.4.22.9. 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.

5.4.22.10. vm-create

vm-create name-label=*name for VM* [name-description=*description string for VM*]

Create a new VM with default parameters. Despite its suggestive name, this command does *not* leave you with a running or runnable VM -- it really creates a template. The resultant UUID must be used as the *template-uuid* argument passed to the **vm-install** command.

This is an advanced command and should *not* normally be used to install VMs. Instead, use the pre-defined templates and the **vm-install** command to instantiate new VMs.

5.4.22.11. 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**.

5.4.22.12. 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* field 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.

5.4.22.13. 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.

5.4.22.14. 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.

5.4.22.15. 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 Section 5.4.22.16, “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.

5.4.22.16. 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).

5.4.22.17. 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.

5.4.22.18. 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 overhead, and is separate 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).

5.4.22.19. 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.

5.4.22.20. 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.

5.4.22.21. 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.

5.4.22.22. 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.

5.4.22.23. 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.

5.4.22.24. 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.

5.4.22.25. 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.

5.4.22.26. 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.

5.4.22.27. vm-vcpu-hotplug

`vm-vcpu-hotplug` 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.

5.4.22.28. vm-vif-list

`vm-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 6. 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.

6.1. 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 Section 5.4.5.2, “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 Section 5.4.5.4, “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.

6.1.1. 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.

Procedure 6.1. 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.)

6.2. 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.

- 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.

6.3. Special debug boot options

Safe mode and serial mode are special boot options that are made available for debugging boot problems. They should not be used in normal operation, but only under explicit instruction from the appropriate Support organization. Both debug options require use of a serial console.

Index

A

AMD-V (AMD hardware virtualization), 3

C

CD commands, xe CLI, 53
CLI (see command line interface)
Command line interface (CLI)
  basic xe syntax, 47
  Bonding commands, 52
  CD commands, 53
  command types, 49
  console commands, 54
  event commands, 54
  host (XenServer Host) commands, 55
  log commands, 63
  low-level list commands, 51
  low-level parameter commands, 50
  network commands, 64
  overview,
  parameter types, 50
  patch commands, 65
  PBD commands, 66
  PIF commands, 67
  Resource pool commands, 69
  shorthand xe syntax, 48
  special characters and syntax, 48
  Storage Manager commands, 71
  Storage repository (SR) commands, 72
  task commands, 74
  Template commands, 75
  update commands, 80
  user commands, 80
  VBD commands, 80
  VDI commands, 83
  VIF commands, 85
  VLAN commands, 87
  VM commands, 87
  xe command reference,
Console commands, xe CLI, 54
Constraints on XenServer Hosts joining resource pool, 3
Creating a resource pool, 4

E

Event commands, xe CLI, 54

F

Fibre Channel storage area network (SAN), 15
Filer, NetApp, 16
FlexVol, NetApp, 16

H

Hardware virtualization
  AMD-V, 3
  Intel VT, 3
HBA (see Host bus adapter)
Host (XenServer Host) commands, xe CLI, 55
Host bus adapter, 15

I

Intel VT (Intel hardware virtualization), 3
iSCSI, 14

L

Log commands, xe CLI, 63
Logical Volume Management (LVM), 13, 13
Logs, XenServer Host, 99

M

Machine failures in a resource pool, 7

N

NAS (see Network attached storage (NFS))
NetApp Filer, 16
Network attached storage (NFS), 13
Network bonding commands, xe CLI, 52
Network commands, xe CLI, 64
Networking VMs,
Networking XenServer Hosts
  modifying hostname, 31

P

Patch commands, xe CLI, 65
PBD commands, xe CLI, 66
PIF commands, xe CLI, 67
Pool commands, xe CLI, 69

Q

QoS settings (Enterprise Edition only)
  virtual disk, 29

R

Removing XenServer Host from a resource pool, 5
Requirements, for creating resource pools, 3
Resource pool, 1,
  constraints on XenServer Hosts joining, 3
  coping with machine failures, 7
  creating, 4
  master, , 7, 8
  member, , 7, 8
  removing XenServer Host from, 5

requirements for creating, 3

S

SAN (see Storage Area Network)
Shared network attached storage (NFS), 13
Shared storage, 4
Storage Area Network, 14
Storage Manager commands, xe CLI, 71
Storage repository (SR)
CD-ROM, 13
commands, xe CLI, 72
DVD-ROM, 13
Fibre Channel storage area network (SAN), 15
local disk, 13
local hotplug devices, 13
NetApp Filer, 16
overview, 11
shared iSCSI storage area network (SAN), 14
shared network attached storage (NFS), 13
USB read/write device, 13

T

Task commands, xe CLI, 74
Template commands, xe CLI, 75
Troubleshooting
special debug boot options, 100
xen-bugtool, ,
XenServer Host logs, 99, 99

U

Update commands, xe CLI, 80
User commands, xe CLI, 80

V

VBD commands, xe CLI, 80
VDI commands, xe CLI, 83
VIF (virtual interface), 31
VIF commands, xe CLI, 85
Virtual disk QoS settings (Enterprise Edition only), 29
Virtual network, 31
VLAN commands, xe CLI, 87
VM
commands, xe CLI, 87
networking,
Virtual disk QoS settings (Enterprise Edition only), 29
VT (Intel hardware virtualization), 3

X

xe command reference,
basic xe syntax, 47
Bonding commands, 52
CD commands, 53
command types, 49
console commands, 54
event commands, 54
host (XenServer Host) commands, 55
log commands, 63
low-level list commands, 51
low-level parameter commands, 50
network commands, 64
parameter types, 50
patch commands, 65
PBD commands, 66
PIF commands, 67
Resource pool commands, 69
shorthand xe syntax, 48
special characters and syntax, 48
Storage Manager commands, 71
Storage repository (SR) commands, 72
task commands, 74
Template commands, 75
update commands, 80
user commands, 80
VBD commands, 80
VDI commands, 83
VIF commands, 85
VLAN commands, 87
VM commands, 87
xe commands, command line interface (CLI),
xen-bugtool, , 99
XenServer Host
constraints, on joining resource pool, 3
joining in resource pools,
logs, 99
networks,
requirements, for joining resource pools, 3
Resource pools, 1
special debug boot options, 100
XenServer Host troubleshooting
logs, 99
special debug boot options, 100
xen-bugtool, , 99