# Xen Cloud Platform Administrator's Guide

## Release 0.1

### 0.1

**Published October 2009**

**1.0 Edition**

# Xen Cloud Platform Administrator's Guide: Release 0.1

Published October 2009

# Contents

# Chapter 1. Document Overview

This document is a system administrator's guide to Xen Cloud Platform™, the platform virtualization solution from Xen.org®. It describes the tasks involved in configuring a Xen Cloud Platform deployment -- in particular, how to set up storage, networking and resource pools, and how to administer Xen Cloud Platform 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:

- Xen Cloud Platform hosts and resource pools
- Xen Cloud Platform storage configuration
- Xen Cloud Platform network configuration
- Xen Cloud Platform workload balancing
- Xen Cloud Platform backup and recovery
- Monitoring and managing Xen Cloud Platform
- Xen Cloud Platform command line interface
- Xen Cloud Platform troubleshooting
- Xen Cloud Platform resource allocation guidelines

## How this Guide relates to other documentation

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

- *Xen Cloud Platform Installation Guide* provides a high level overview of Xen Cloud Platform, along with step-by-step instructions on installing Xen Cloud Platform hosts.
- *Xen Cloud Platform Virtual Machine Installation Guide* describes how to install Linux and Windows VMs on top of a Xen Cloud Platform deployment. As well as installing new VMs from install media (or using the VM *templates* provided with the Xen Cloud Platform release), this guide also explains how to create VMs from existing physical machines, using a process called *P2V*.
- *Xen Cloud Platform Software Development Kit Guide* presents an overview of the Xen Cloud Platform SDK -- a selection of code samples that demonstrate how to write applications that interface with Xen Cloud Platform hosts.
- *XenAPI Specification* provides a programmer's reference guide to the Xen Cloud Platform API.
- *Xen Cloud Platform User Security* considers the issues involved in keeping your Xen Cloud Platform installation secure.
- *Release Notes* provides a list of known issues that affect this release.

# Chapter 2. Xen Cloud Platform hosts and resource pools

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.

## Hosts and resource pools overview

A *resource pool* comprises multiple Xen Cloud Platform 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* Xen Cloud Platform host which has sufficient memory and then dynamically moved between Xen Cloud Platform hosts while running with minimal downtime (XenMotion). If an individual Xen Cloud Platform host suffers a hardware failure, then the administrator can restart the failed VMs on another Xen Cloud Platform host in the same resource pool. If high availability (HA) is enabled on the resource pool, VMs will automatically be moved if their host fails. Up to 16 hosts are supported per resource pool, although this restriction is not enforced.

A pool always has at least one physical node, known as the *master*. Only the master node exposes an administration interface (used by XenCenter and the CLI); the master forwards commands to individual members as necessary.

## Requirements for creating resource pools

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

- the CPUs on the server joining the pool are the same (in terms of vendor, model, and features) as the CPUs on servers already in the pool.
- the server joining the pool is running the same version of Xen Cloud Platform software, at the same patch level, as servers already in the pool.

The software will enforce additional constraints when joining a server to a pool – in particular:

- it is not a member of an existing resource pool
- it has no shared storage configured
- there are no running or suspended VMs on the Xen Cloud Platform host which is joining
- there are no active operations on the VMs in progress, such as one shutting down

You must also check that the clock of the host joining the pool is synchronized to the same time as the pool master (for example, by using NTP), that its management interface is not bonded (you can configure this once the host has successfully joined the pool), and that

its management IP address is static (either configured on the host itself or by using an appropriate configuration on your DHCP server).

Xen Cloud Platform hosts in resource pools may contain different numbers of physical network interfaces and have local storage repositories 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 by passing a `--force` parameter.

---

**Note**

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

---

Although not a strict technical requirement for creating a resource pool, the advantages of pools (for example, the ability to dynamically choose on which Xen Cloud Platform host to run a VM and to dynamically move a VM between Xen Cloud Platform hosts) are only available if the pool has one or more shared storage repositories. If possible, postpone creating a pool of Xen Cloud Platform hosts until shared storage is available. Once shared storage has been added, Xen.org recommends that you move existing VMs whose disks are in local storage into shared storage. This can be done using the **xe vm-copy** command.

## Creating a resource pool

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

- VM, local, and remote storage configuration is added to the pool-wide database. All of these will still be tied to the joining host in the pool unless you explicitly take action to make the resources shared after the join has completed.
- The joining host inherits existing shared storage repositories in the pool and appropriate PBD records are created so that the new host can access existing shared storage automatically.
- Networking information is partially inherited to the joining host: the *structural* details of NICs, VLANs and bonded interfaces are all inherited, but *policy* information is not. This policy information, which must be re-configured, includes:
  - the IP addresses of management NICs, which are preserved from the original configuration
  - the location of the management interface, which remains the same as the original configuration. For example, if the other pool hosts have their management interface on a bonded interface, then the joining host must be explicitly migrated to the bond once it has joined. See To add NIC bonds to the pool master and other hosts for details on how to migrate the management interface to a bond.
  - Dedicated storage NICs, which must be re-assigned to the joining host from the CLI, and the PBDs re-plugged to route the traffic accordingly. This is because IP addresses are not assigned as part of the pool join operation, and the storage NIC is not useful

without this configured correctly. See the section called “Configuring a dedicated storage NIC” for details on how to dedicate a storage NIC from the CLI.

### To join Xen Cloud Platform hosts *host1* and *host2* into a resource pool using the CLI

1. Open a console on Xen Cloud Platform host *host2*.
2. Command Xen Cloud Platform host *host2* to join the pool on Xen Cloud Platform host *host1* by issuing the command:

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

The *master-address* must be set to the fully-qualified domain name of Xen Cloud Platform host *host1* and the *password* must be the administrator password set when Xen Cloud Platform host *host1* was installed.

### Naming a resource pool

- Xen Cloud Platform hosts belong to an unnamed pool by default. To create your first resource pool, rename the existing nameless pool. You can use tab-complete to get the <*pool_uuid*>:

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

## Adding shared storage

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

### Adding NFS shared storage to a resource pool using the CLI

1. Open a console on any Xen Cloud Platform 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 Xen Cloud Platform host in the pool and any Xen Cloud Platform hosts that subsequently join will also be connected to the storage. The Universally Unique Identifier (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. See Chapter 3, *Storage* for information about creating other types of shared storage.

# Installing and managing VMs on shared storage

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

## Installing a Debian Etch (4.0) VM

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

```
xe sr-list
```

3. Create the Debian VM by issuing the command

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

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

4. Start the Debian VM with the command

```
xe vm-start vm=<etch>
```

The master will choose a Xen Cloud Platform host from the pool to start the VM. If the *on* parameter is provided, the VM will start on the specified Xen Cloud Platform host. If the requested Xen Cloud Platform host is unable to start the VM, the command will fail. To request that a VM is always started on a particular Xen Cloud Platform host, set the *affinity* parameter of the VM to the UUID of the desired Xen Cloud Platform 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 Xen Cloud Platform hosts.

5. You can use XenMotion to move the Debian VM to another Xen Cloud Platform host with the command

```
xe vm-migrate vm=<etch> host=<host_name> --live
```

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

### Note

When a VM is migrated, the domain on the original hosting server is destroyed and the memory that VM used is zeroed out before Xen makes it available to new VMs. This ensures that there is no information leak from old VMs to new ones. As a conse-

quence, it is possible that sending multiple near-simultaneous commands to migrate a number of VMs, when near the memory limit of a server (for example, a set of VMs consuming 3GB migrated to a server with 4GB of physical memory), the memory of an old domain might not be scrubbed before a migration is attempted, causing the migration to fail with a `HOST_NOT_ENOUGH_FREE_MEMORY` error. Inserting a delay between migrations should allow Xen the opportunity to successfully scrub the memory and return it to general use.

## Removing a Xen Cloud Platform host from a resource pool

When a Xen Cloud Platform 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 Xen Cloud Platform host from a pool if there is important data on the local disks.

### To remove a host from a resource pool using the CLI

1. Open a console on any host in the pool.
2. Find the UUID of the host *b* using the command

```
xe host-list
```

3. Eject the host from the pool:

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

The Xen Cloud Platform 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 using the **xe vm-copy** CLI command.

When a Xen Cloud Platform 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 Xen Cloud Platform 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 Xen Cloud Platform 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 Xen Cloud Platform hosts can be ejected (or physically fail) without loss of data.

## Authenticating users using Active Directory (AD)

Xen Cloud Platform supports the authentication of users through AD. This makes it easier to control access to Xen Cloud Platform hosts. Active Directory users can use the xe CLI (passing appropriate `-u` and `-pw` arguments). Authentication is done on a per-resource pool basis.

Access is controlled by the use of *subjects*. A subject in Xen Cloud Platform maps to an entity on your directory server (either a user or a group). When external authentication is enabled, the credentials used to create a session are first checked against the local root credentials (in case your directory server is unavailable) and then against the subject list. To permit access, you must create a subject entry for the person or group you wish to grant access to. This can be done using the xe CLI.

## Configuring Active Directory authentication

Xen Cloud Platform supports use of Active Directory servers using Windows 2003 or later.

For external authentication using Active Directory to be successful, it is important that the clocks on your Xen Cloud Platform hosts are synchronized with those on your Active Directory server. When Xen Cloud Platform joins the Active Directory domain, this will be checked and authentication will fail if there is too much skew between the servers.

**Note**

The servers can be in different time-zones, and it is the UTC time that is compared. To ensure synchronization is correct, you may choose to use the same NTP servers for your Xen Cloud Platform pool and the Active Directory server.

Active Directory authentication for a Xen Cloud Platform host requires that the same DNS servers are used for both the Active Directory server (configured to allow for interoperability) and the Xen Cloud Platform host. In some configurations, the active directory server may provide the DNS itself. This can be achieved either using DHCP to provide the IP address and a list of DNS servers to the Xen Cloud Platform host, or by setting values in the PIF objects or using the installer if a manual static configuration is used.

Xen.org recommends enabling DCHP to broadcast host names. In particular, the host names `localhost` or `linux` should not be assigned to hosts. Host names must consist solely of no more than 156 alphanumeric characters, and may not be purely numeric.

### Enabling external authentication on a pool

- External authentication using Active Directory can be configured using the CLI using the command below.

```
xe pool-enable-external-auth auth-type=AD \
       service-name=<full-qualified-domain> \
       config:user=<username> \
       config:pass=<password>
```

The user specified needs to have `Add/remove computer objects or workstations` privileges, which is the default for domain administrators.

**Note**

If you are not using DHCP on the network that Active Directory and your Xen Cloud Platform hosts use you can use these two approaches to setup your DNS:

1. Configure the DNS server to use on your Xen Cloud Platform hosts:

```
xe pif-reconfigure-ip mode=static dns=<dnshost>
```

2. Manually set the management interface to use a PIF that is on the same network as your DNS server:

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

---

**Note**

External authentication is a per-host property. However, Xen.org advises that you enable and disable this on a per-pool basis – in this case Xen Cloud Platform will deal with any failures that occur when enabling authentication on a particular host and perform any roll-back of changes that may be required, ensuring that a consistent configuration is used across the pool. Use the **host-param-list** command to inspect properties of a host and to determine the status of external authentication by checking the values of the relevant fields.

---

### Disabling external authentication

- Use the following xe commnad to disable Active Directory authentication:

```
xe pool-disable-external-auth
```

## User authentication

To allow a user access to your Xen Cloud Platform host, you must add a subject for that user or a group that they are in. (Transitive group memberships are also checked in the normal way, for example: adding a subject for group A, where group A contains group B and `user 1` is a member of group B would permit access to `user 1`.) If you wish to manage user permissions in Active Directory, you could create a single group that you then add and remove users to/from; alternatively, you can add and remove individual users from Xen Cloud Platform, or a combination of users and groups as your would be appropriate for your authentication requirements. The subject list can be managed from the CLI as described below.

When authenticating a user, the credentials are first checked against the local root account, allowing you to recover a system whose AD server has failed. If the credentials (i.e. user-name then password) do not match/authenticate, then an authentication request is made to the AD server – if this is successful the user's information will be retrieved and validated against the local subject list, otherwise access will be denied. Validation against the subject list will succeed if the user or a group in the transitive group membership of the user is in the subject list.

### Allowing a user access to Xen Cloud Platform using the CLI

- To add an AD subject to Xen Cloud Platform:

```
xe subject-add subject-name=<entity name>
```

The entity name should be the name of the user or group to which you want to grant access. You may optionally include the domain of the entity (for example, '*<xendt\user1>*' as opposed to '*<user1>*') although the behavior will be the same unless disambiguation is required.

### Removing access for a user using the CLI

1. Identify the subject identifier for the subject you wish to revoke access. This would be the user or the group containing the user (removing a group would remove access to all users in that group, providing they are not also specified in the subject list). You can do this using the subject list command:

```
xe subject-list
```

   You may wish to apply a filter to the list, for example to get the subject identifier for a user named `user1` in the `testad` domain, you could use the following command:

```
xe subject-list other-config:subject-name='<domain\user>'
```

2. Remove the user using the **subject-remove** command, passing in the subject identifier you learned in the previous step:

```
xe subject-remove subject-identifier=<subject identifier>
```

3. You may wish to terminate any current session this user has already authenticated. See Terminating all authenticated sessions using xe and Terminating individual user sessions using xe for more information about terminating sessions. If you do not terminate sessions the users whose permissions have been revoked may be able to continue to access the system until they log out.

### Listing subjects with access

- To identify the list of users and groups with permission to access your Xen Cloud Platform host or pool, use the following command:

```
xe subject-list
```

## Removing access for a user

Once a user is authenticated, they will have access to the server until they end their session, or another user terminates their session. Removing a user from the subject list, or removing them from a group that is in the subject list, will not automatically revoke any already-authenticated sessions that the user has; this means that they may be able to continue to access the pool using other API sessions that they have already created. In order to terminate these sessions forcefully, the CLI provide facilities to terminate individual sessions, or all currently active sessions. See below for procedures using the CLI.

### Terminating all authenticated sessions using xe

- Execute the following CLI command:

```
xe session-subject-identifier-logout-all
```

### Terminating individual user sessions using xe

1. Determine the subject identifier whose session you wish to log out. Use either the **session-subject-identifier-list** or **subject-list** xe commands to find this (the first shows users who have sessions, the second shows all users but can be filtered, for example, using a command like **xe subject-list other-config:subject-name=xendt\\user1** – depending on your shell you may need a double-backslash as shown).
2. Use the **session-subject-logout** command, passing the subject identifier you have determined in the previous step as a parameter, for example:

```
xe session-subject-identifier-logout subject-identifier=<subject-id>
```

## Leaving an AD domain

Run the **pool-disable-external-auth** command, specifying the pool uuid if required.

---

**Note**

Leaving the domain will not cause the host objects to be removed from the AD database. See this knowledge base article for more information about this and how to remove the disabled host entries.

---

# 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 Xen Cloud Platform 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, generating snapshots for backup purposes and some best practices for managing storage in Xen Cloud Platform host environments. Finally, the virtual disk QoS (quality of service) settings are described.

## Storage Overview

This section explains what the Xen Cloud Platform storage objects are and how they are related to each other.

### Storage Repositories (SRs)

Xen Cloud Platform 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. The Xen Cloud Platform SR is very flexible, with built-in support for IDE, SATA, SCSI and SAS drives locally connected, and iSCSI, NFS, SAS and Fibre Channel remotely connected. The SR and VDI abstractions allow 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 Xen Cloud Platform host can use multiple SRs and different SR types simultaneously. These SRs can be shared between hosts or dedicated to particular hosts. Shared storage is pooled between multiple hosts within a defined resource pool. A shared SR must be network accessible to each host. All hosts in a single resource pool must have at least one shared SR in common.

SRs are storage targets containing virtual disk images (VDIs). SR commands provide operations for creating, destroying, resizing, cloning, connecting and discovering the individual VDIs that they contain.

A storage repository is a persistent, on-disk data structure. For SR types that use an underlying block device, the process of creating a new SR involves erasing any existing data on the specified storage target. Other storage types such as NFS, Netapp, Equallogic and StorageLink SRs, create a new container on the storage array in parallel to existing SRs.

CLI operations to manage storage repositories are described in the section called “SR commands”.

## Virtual Disk Images (VDIs)

Virtual Disk Images are a storage abstraction that is presented to a VM. VDIs are the fundamental unit of virtualized storage in Xen Cloud Platform. Similar to SRs, VDIs are persistent, on-disk objects that exist independently of Xen Cloud Platform hosts. CLI operations to manage VDIs are described in the section called “VDI commands”. The actual on-disk representation of the data differs by the SR type and is managed by a separate storage plugin interface for each SR, called the SM API.

## Physical Block Devices (PBDs)

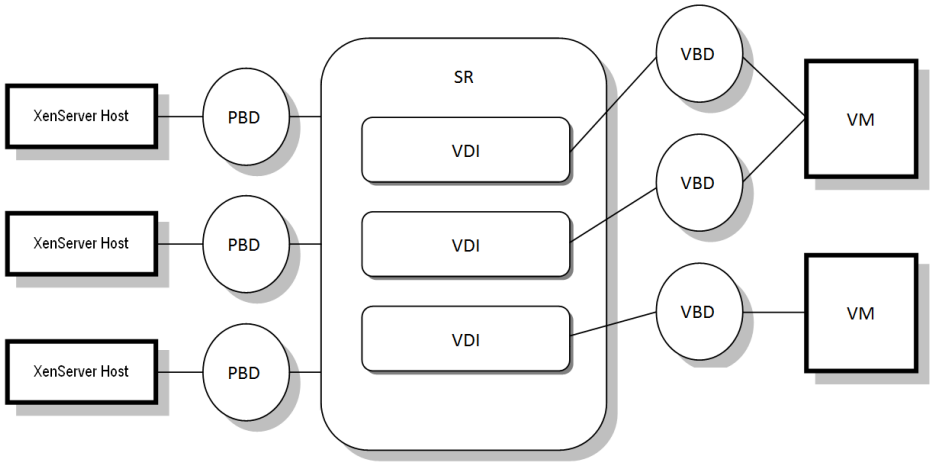
Physical Block Devices represent the interface between a physical server and an attached SR. PBDs are connector objects that allow a given SR to be mapped to a Xen Cloud Platform host. PBDs store the device configuration fields that are used to connect to and interact with a given storage target. For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

## Virtual Block Devices (VBDs)

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

## Summary of Storage objects

The following image is a summary of how the storage objects presented so far are related:




***Graphical overview of storage repositories and related objects***

## Virtual Disk Data Formats

In general, there are three types of mapping of physical storage to a VDI:

- *File-based VHD on a Filesystem;* VM images are stored as thin-provisioned VHD format files on either a local non-shared Filesystem (EXT type SR) or a shared NFS target (NFS type SR)
- *Logical Volume-based VHD on a LUN;* The default XenServer blockdevice-based storage inserts a Logical Volume manager on a disk, either a locally attached device (LVM type SR) or a SAN attached LUN over either Fibre Channel (LVMoHBA type SR), iSCSI (LVMoISCSI type SR) or SAS (LVMoHBA type Sr). VDIs are represented as volumes within the Volume manager and stored in VHD format to allow thin provisioning of reference nodes on snapshot and clone.
- *LUN per VDI;* LUNs are directly mapped to VMs as VDIs by SR types that provide an array-specific plugin (Netapp, Equallogic or StorageLink type SRs). The array storage abstraction therefore matches the VDI storage abstraction for environments that manage storage provisioning at an array level.

### VHD-based VDIs

VHD files may 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 (CoW) 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.

The VHD format used by LVM-based and File-based SR types in Xen Cloud Platform uses sparse provisioning. The image file is automatically extended in 2MB chunks as the VM writes data into the disk. For File-based VHD, this has the considerable benefit that VM image files take up only as much space on the physical storage as required. With LVM-based VHD the underlying logical volume container must be sized to the virtual size of the VDI, however unused space on the underlying CoW instance disk is reclaimed when a snapshot or clone occurs. The difference between the two behaviors can be characterized in the following way:

- For *LVM-based VHDs*, the difference disk nodes within the chain consume only as much data as has been written to disk but the leaf nodes (VDI clones) remain fully inflated to the virtual size of the disk. Snapshot leaf nodes (VDI snapshots) remain deflated when not in use and can be attached Read-only to preserve the deflated allocation. Snapshot nodes that are attached Read-Write will be fully inflated on attach, and deflated on detach.
- For *file-based VHDs*, all nodes consume only as much data as has been written, and the leaf node files grow to accommodate data as it is actively written. 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.

When cloning VMs based off a single VHD template, each child VM forms a chain where new changes are written to the new VM, and old blocks are directly read from the parent template. If the new VM was converted into a further template and more VMs cloned, then the resulting chain will result in degraded performance. Xen Cloud Platform supports a maximum chain length of 30, but it is generally not recommended that you approach this

limit without good reason. If in doubt, you can always "copy" the VM using Xen Cloud Platform or the **vm-copy** command, which resets the chain length back to 0.

### VHD Chain Coalescing

VHD images support chaining, which is the process whereby information shared between one or more VDIs is not duplicated. This leads to a situation where trees of chained VDIs are created over time as VMs and their associated VDIs get cloned. When one of the VDIs in a chain is deleted, Xen Cloud Platform rationalizes the other VDIs in the chain to remove unnecessary VDIs.

This coalescing process runs asynchronously. The amount of disk space reclaimed and the time taken to perform the process depends on the size of the VDI and the amount of shared data. Only one coalescing process will ever be active for an SR. This process thread runs on the SR master host.

If you have critical VMs running on the master server of the pool and experience occasional slow IO due to this process, you can take steps to mitigate against this:

- Migrate the VM to a host other than the SR master
- Set the disk IO priority to a higher level, and adjust the scheduler. See the section called “Virtual disk QoS settings” for more information.

### Space Utilisation

Space utilisation is always reported based on the current allocation of the SR, and may not reflect the amount of virtual disk space allocated. The reporting of space for LVM-based SRs versus File-based SRs will also differ given that File-based VHD supports full thin provisioning, while the underlying volume of an LVM-based VHD will be fully inflated to support potential growth for writeable leaf nodes. Space utilisation reported for the SR will depend on the number of snapshots, and the amount of difference data written to a disk between each snapshot.

LVM-based space utilisation differs depending on whether an LVM SR is upgraded or created as a new SR in Xen Cloud Platform. Upgraded LVM SRs will retain a base node that is fully inflated to the size of the virtual disk, and any subsequent snapshot or clone operations will provision at least one additional node that is fully inflated. For new SRs, in contrast, the base node will be deflated to only the data allocated in the VHD overlay.

When VHD-based VDIs are deleted, the space is marked for deletion on disk. Actual removal of allocated data may take some time to occur as it is handled by the coalesce process that runs asynchronously and independently for each VHD-based SR.

## LUN-based VDIs

Mapping a raw LUN as a Virtual Disk image is typically the most high-performance storage method. For administrators that want to leverage existing storage SAN infrastructure such as Netapp, Equallogic or StorageLink accessible arrays, the array snapshot, clone and thin provisioning capabilities can be exploited directly using one of the array specific adapter SR types (Netapp, Equallogic or StorageLink). The virtual machine storage operations are mapped directly onto the array APIs using a LUN per VDI representation. This includes

activating the data path on demand such as when a VM is started or migrated to another host.

Managed NetApp LUNs are accessible using 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 using the Xen Cloud Platform host management framework.

EqualLogic storage is accessible using the EqualLogic SR driver type, and is hosted on an EqualLogic storage array running a firmware version of 4.0 or greater. LUNs are allocated and mapped dynamically to the host using the Xen Cloud Platform host management framework.

For further information on StorageLink supported array systems and the various capabilities in each case, please refer to the StorageLink documentation directly.

# Storage configuration

This section covers creating storage repository types and making them available to a Xen Cloud Platform host. The examples provided pertain to storage configuration using the CLI, which provides the greatest flexibility.

## Creating Storage Repositories

This section explains how to create Storage Repositories (SRs) of different types and make them available to a Xen Cloud Platform host. The examples provided cover creating SRs using the xe CLI.

---

**Note**

Local SRs of type `lvm` and `ext` can only be created using the xe CLI. After creation all SR types can be managed by the xe CLI.

---

There are two basic steps involved in creating a new storage repository for use on a Xen Cloud Platform host using the CLI:

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

These steps differ in detail depending on the type of SR being created. In all examples the **sr-create** command returns the UUID of the created 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 Xen Cloud Platform host and attach it to another. See the section called “Destroying or forgetting a SR” for details.

## Upgrading LVM storage from Xen Cloud Platform 5.0 or earlier

See the *Xen Cloud Platform Installation Guide* for information on upgrading LVM storage to enable the latest features. Local, LVM on iSCSI, and LVM on HBA storage types from older

(Xen Cloud Platform 5.0 and before) product versions will need to be upgraded before they will support snapshot and fast clone.

**Note**

Upgrade is a one-way operation so Xen.org recommends only performing the upgrade when you are certain the storage will no longer need to be attached to a pool running an older software version.

## LVM performance considerations

The snapshot and fast clone functionality provided in Xen Cloud Platform 5.5 and later for LVM-based SRs comes with an inherent performance overhead. In cases where optimal performance is desired, Xen Cloud Platform supports creation of VDIs in the *raw* format in addition to the default VHD format. The Xen Cloud Platform snapshot functionality is not supported on raw VDIs.

**Note**

Non-transportable snapshots using the default Windows VSS provider will work on any type of VDI.

**Warning**

Do not try to snapshot a VM that has `type=raw` disks attached. This could result in a partial snapshot being created. In this situation, you can identify the orphan snapshot VDIs by checking the `snapshot-of` field and then deleting them.

### VDI types

In general, VHD format VDIs will be created. You can opt to use raw at the time you create the VDI; this can only be done using the xe CLI. After software upgrade from a previous Xen Cloud Platform version, existing data will be preserved as backwards-compatible raw VDIs but these are special-cased so that snapshots can be taken of them once you have allowed this by upgrading the SR. Once the SR has been upgraded and the first snapshot has been taken, you will be accessing the data through a VHD format VDI.

To check if an SR has been upgraded, verify that its `sm-config:use_vhd` key is `true`. To check if a VDI was created with `type=raw`, check its `sm-config` map. The **sr-param-list** and **vdi-param-list** xe commands can be used respectively for this purpose.

### Creating a raw virtual disk using the xe CLI

1. Run the following command to create a VDI given the UUID of the SR you want to place the virtual disk in:

```
xe vdi-create sr-uuid=<sr-uuid> type=user virtual-size=<virtual-size> \ name-label=<VDI nam
```

2. Attach the new virtual disk to a VM and use your normal disk tools within the VM to partition and format, or otherwise make use of the new disk. You can use the **vbd-create** command to create a new VBD to map the virtual disk into your VM.

## Converting between VDI formats

It is not possible to do a direct conversion between the raw and VHD formats. Instead, you can create a new VDI (either raw, as described above, or VHD if the SR has been upgraded or was created on Xen Cloud Platform 5.5 or later) and then copy data into it from an existing volume. Xen.org recommends that you use the xe CLI to ensure that the new VDI has a virtual size at least as big as the VDI you are copying from (by checking its virtual-size field, for example by using the **vdi-param-list** command). You can then attach this new VDI to a VM and use your preferred tool within the VM (standard disk management tools in Windows, or the **dd** command in Linux) to do a direct block-copy of the data. If the new volume is a VHD volume, it is important to use a tool that can avoid writing empty sectors to the disk so that space is used optimally in the underlying storage repository — in this case a file-based copy approach may be more suitable.

## Probing an SR

The **sr-probe** 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 the **sr-probe** command 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 XML.

For example, a known iSCSI target can be probed by specifying its name or IP address, and the set of IQNs available on the target will be returned:

```
xe sr-probe type=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 returns 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>
            14945540000000000000000002000000b70200000f000000
        </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=14945540000000000000000002000000b70200000f000000

<?xml version="1.0" ?>
<SRlist>
    <SR>
        <UUID>
            3f6e1ebd-8687-0315-f9d3-b02ab3adc4a6
        </UUID>
        <Devlist>
            /dev/disk/by-id/scsi-14945540000000000000000002000000b70200000f000000
        </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 |

| SR type | device-config parameter, in order of dependency | Can be probed? | Required for sr-create? |
|---|---|---|---|
| | SCSIid | Yes | Yes |
| lvmohba | SCSIid | Yes | Yes |
| netapp | target | No | Yes |
| | 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 |
| equallogic | target | No | Yes |
| | username | No | Yes |
| | password | No | Yes |
| | chapuser | No | No |
| | chappassword | No | No |
| | storagepool | No† | Yes |
| cslg | target | No | Yes |

| SR type | device-config parameter, in order of dependency | Can be probed? | Required for sr-create? |
|---|---|---|---|
| | storageSystemId | Yes | Yes |
| | storagePoolId | Yes | Yes |
| | username | No | No ‡ |
| | password | No | No ‡ |
| | cslport | No | No ‡ |
| | chapuser | No | No ‡ |
| | chappassword | No | No ‡ |
| | provision-type | Yes | No |
| | protocol | Yes | No |
| | provision-options | Yes | No |
| | raid-type | Yes | No |

*Aggregate probing is only possible at **sr-create** time. It needs to be done there so that the aggregate can be specified at the point that the SR is created.

†Storage pool probing is only possible at **sr-create** time. It needs to be done there so that the aggregate can be specified at the point that the SR is created.

‡If the username, password, or port configuration of the StorageLink service are changed from the default value then the appropriate parameter and value must be specified.

## Storage Multipathing

Dynamic multipathing support is available for Fibre Channel and iSCSI storage backends. By default, it uses round-robin mode load balancing, so both routes have active traffic on them during normal operation. You can enable multipathing in the xe CLI.

### Caution

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

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

### To enable storage multipathing using the xe CLI

1. Unplug all PBDs on the host:

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

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

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

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

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

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

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

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

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

Multipath support in Xen Cloud Platform is based on the device-mapper `multipathd` components. Activation and deactivation of multipath nodes is handled automatically by the Storage Manager API. Unlike the standard `dm-multipath` tools in linux, device mapper nodes are not automatically created for all LUNs on the system, and it is only when LUNs are actively used by the storage management layer that new device mapper nodes are provisioned. It is unnecessary therefore to use any of the `dm-multipath` CLI tools to query or refresh DM table nodes in Xen Cloud Platform. Should it be necessary to query the status of device-mapper tables manually, or list active device mapper multipath nodes on the system, use the `mpathutil` utility:

- mpathutil list
- mpathutil status

Unlike the standard `dm-multipath` tools in Linux, device mapper nodes are not automatically created for all LUNs on the system. As LUNs are actively used by the storage management layer, new device mapper nodes are provisioned. It is unnecessary to use any of the `dm-multipath` CLI tools to query or refresh DM table nodes in Xen Cloud Platform.

**Note**

Due to incompatibilities with the integrated multipath management architecture, the standard `dm-multipath` CLI utility *should not be used* with Xen Cloud Platform. Please use the `mpathutil` CLI tool for querying the status of nodes on the host.

**Note**

Multipath support in Equallogic arrays does not encompass Storage IO multipathing in the traditional sense of the term. Multipathing must be handled at the network/NIC bond level.

Refer to the Equallogic documentation for information about configuring network failover for Equallogic SRs/LVMoISCSI SRs.

## Storage Repository Types

The storage repository types supported in Xen Cloud Platform are provided by plugins in the control domain; these can be examined and plugins supported third parties can be added to the `/opt/xensource/sm` directory. Modification of these files is unsupported, but visibility of these files may be valuable to developers and power users. New storage manager plugins placed in this directory are automatically detected by Xen Cloud Platform. Use the **sm-list** command (see the section called “Storage Manager commands”) to list the available SR types .

New storage repositories are created using the **sr-create** command. This command creates a new SR on the storage substrate (potentially destroying any existing data), and creates the SR API object and a corresponding PBD record, enabling VMs to use the storage. On successful creation of the SR, the PBD is automatically plugged. If the SR `shared=true` flag is set, a PBD record is created and plugged for every Xen Cloud Platform Host in the resource pool.

All Xen Cloud Platform SR types support VDI resize, fast cloning and snapshot. SRs based on the LVM SR type (local, iSCSI, or HBA) provide thin provisioning for snapshot and hidden parent nodes. The other SR types support full thin provisioning, including for virtual disks that are active.

### Note

Automatic LVM metadata archiving is disabled by default. This does not prevent metadata recovery for LVM groups.

### Warning

When VHD VDIs are not attached, for example in the case of a VDI snapshot, they are stored by default thinly-provisioned. Because of this it is imperative to ensure that there is sufficient disk-space available for the VDI to become thickly provisioned when attempting to attach it. VDI clones, however, are thickly-provisioned.

The maximum supported VDI sizes are:

| Storage type | Maximum VDI size |
|---|---|
| EXT3 | 2TB |
| LVM | 2TB |
| Netapp | 2TB |
| EqualLogic | 15TB |
| ONTAP(NetApp) | 12TB |

## Local LVM

The Local LVM type presents disks within a locally-attached Volume Group.

By default, Xen Cloud Platform 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. A VDI is implemented in VHD format in an LVM logical volume of the specified size.

Xen Cloud Platform versions prior to 0.1 did not use the VHD format and will remain in legacy mode. See the section called “Upgrading LVM storage from Xen Cloud Platform 5.0 or earlier” for information about upgrading a storage repository to the new format.

### 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
```

## Local EXT3 VHD

The Local EXT3 VHD type represents disks as VHD files stored on a local path.

Local disks can also be configured with a local EXT SR to serve VDIs stored in the VHD format. Local disk EXT SRs must be configured using the Xen Cloud Platform CLI.

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

### 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
```

## udev

The udev type represents devices plugged in using the udev device manager as VDIs.

Xen Cloud Platform 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 Xen Cloud Platform host. The other is for a USB device plugged into a USB port of the Xen Cloud Platform host. VDIs that represent the media come and go as disks or USB sticks are inserted and removed.

## ISO

The ISO type handles CD images stored as files in ISO format. This SR type is useful for creating shared ISO libraries.

## EqualLogic

The EqualLogic SR type maps LUNs to VDIs on an EqualLogic array group, allowing for the use of fast snapshot and clone features on the array.

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

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

---

**Warning**

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

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

---

### Creating a shared EqualLogic SR

Device-config parameters for EqualLogic SRs:

| Parameter Name | Description | Optional? |
|---|---|---|
| target | the IP address or hostname of the EqualLogic array that hosts the SR | no |

| Parameter Name | Description | Optional? |
|---|---|---|
| username | the login username used to manage the LUNs on the array | no |
| password | the login password used to manage the LUNs on the array | no |
| storagepool | the storage pool name | no |
| chapuser | the username to be used for CHAP authentication | yes |
| chappassword | the password to be used for CHAP authentication | yes |
| allocation | specifies whether to use thick or thin provisioning. Default is thick. Thin provisioning reserves a minimum of 10% of volume space. | yes |
| snap-reserve-percentage | sets the amount of space, as percentage of volume reserve, to allocate to snapshots. Default is 100%. | yes |
| snap-depletion | sets the action to take when snapshot reserve space is exceeded. *volume-offline* sets the volume and all its snapshots offline. This is the default action. The *delete-oldest* action deletes the oldest snapshot until enough space is available for creating the new snapshot. | yes |

Use the **sr-create** command to create an EqualLogic SR. For example:

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

## NetApp

The NetApp type maps LUNs to VDIs on a NetApp server, enabling the use of fast snapshot and clone features on the filer.

**Note**

NetApp and EqualLogic SRs require a **Xen.org Essentials for Xen Cloud Platform** license to use the special integration with the NetApp and Dell EqualLogic SR types, but you can use them as ordinary iSCSI, FC, or NFS storage with free Xen Cloud Platform, without the benefits of direct control of hardware features. To learn more about Xen.org Essentials for Xen Cloud Platform and to find out how to upgrade, visit the Xen.org website here.

If you have access to Network Appliance™ (NetApp) storage 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 Xen Cloud Platform deployment. The Xen Cloud Platform driver uses the ZAPI interface to the storage to create a group of *FlexVols* that correspond to an SR. VDIs are created as virtual LUNs on the storage, and attached to Xen Cloud Platform hosts using an iSCSI data path. There is a direct mapping between a VDI and a raw LUN that does not require any additional volume metadata. 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 storage for data efficiency and performance and to ensure compatibility with existing ONTAP management tools.

See the section called “Creating a shared NetApp SR over iSCSI” for an example of how to create them using the xe CLI.

### FlexVols

NetApp uses FlexVols as the basic unit of manageable data. 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. Because there is a one-to-one mapping of LUNs to VDIs, and because often a VM will have more than one VDI, the resource limitations of a single FlexVol can easily be reached. Also, the act of taking a snapshot includes snapshotting all the LUNs within a FlexVol and the VM clone operation indirectly relies on snapshots in the background as well as the VDI snapshot operation for backup purposes.

There are two constraints to consider when mapping the virtual storage objects of the Xen Cloud Platform host to the physical storage. To maintain space efficiency it makes sense to limit the number of LUNs per FlexVol, yet at the other extreme, to avoid resource limitations a single LUN per FlexVol provides the most flexibility. However, because there is a vendor-imposed limit of 200 or 500 FlexVols, per filer (depending on the NetApp model), this creates a limit of 200 or 500 VDIs per filer and it is therefore important to select a suitable number of FlexVols taking these parameters into account.

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, because there are 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 lists all aggregates available and the unused disk space on each.

**Note**

Aggregate probing is only possible at **sr-create** time so that the aggregate can be specified at the point that the SR is created, but is not probed by the **sr-probe** command.

Xen.org strongly recommends that you configure an aggregate exclusively for use by Xen Cloud Platform storage, because space guarantees and allocation cannot be correctly managed if other applications are sharing the resource.

### Thick or thin provisioning

When creating NetApp storage, you can also choose the type of space management used. By default, allocated space is thickly provisioned to ensure that VMs never run out of disk space and that all virtual allocation guarantees are fully enforced on the filer. Selecting thick provisioning ensures that whenever a VDI (LUN) is allocated on the filer, sufficient space is reserved to guarantee that it will never run out of space and consequently experience failed writes to disk. Due to the nature of the Ontap FlexVol space provisioning algorithms the best practice guidelines for the filer require that at least twice the LUN space is reserved to account for background snapshot data collection and to ensure that writes to disk are never blocked. In addition to the double disk space guarantee, Ontap also requires some additional space reservation for management of unique blocks across snapshots. The guideline on this amount is 20% above the reserved space. 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 administrator to present more storage space to the VMs connecting to the SR than is actually available on the SR. There are no space guarantees, and allocation of a LUN does not claim any data blocks in the FlexVol until the VM writes data. This might 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 might be created and destroyed frequently without ever utilizing the full virtual allocated disk.

**Warning**

If you are using thin provisioning in production environments, take appropriate measures to ensure that you never run out of storage space. VMs attached to storage that is full

will fail to write to disk, and in some cases may fail to read from disk, possibly rendering the VM unusable.

---

#### FAS Deduplication

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

#### Access Control

Because FlexVol operations such as volume creation and volume snapshotting require administrator privileges on the filer itself, Xen.org recommends that the Xen Cloud Platform host is provided with suitable administrator username and password credentials at configuration time. In situations where the Xen Cloud Platform 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 and then introduce the SR to the Xen Cloud Platform host using the **sr-introduce** xe CLI command. Note, however, that operations such as VM cloning or snapshot generation will fail in this situation due to insufficient access privileges.

#### Licenses

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

#### Further information

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

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

### 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 that hosts the SR | no |
| port | the port to use for connecting to the NetApp server that hosts the SR. Default is port 80. | yes |
| usehttps | specifies whether to use a secure TLS-based connection to the NetApp server that hosts the SR [true\|false]. Default is false. | yes |
| username | the login username used to manage the LUNs on the filer | no |
| password | the login password used to manage 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 to each SR | yes |
| chapuser | the username for CHAP authentication | yes |
| chappassword | the password for CHAP authentication | yes |
| allocation | specifies whether to provision LUNs using thick or thin provisioning. Default is thick | yes |
| asis | specifies whether to use FAS Deduplication if available. Default is false | yes |

Setting the SR *`other-config:multiplier`* parameter to a valid value adjusts the default multiplier attribute. By default Xen Cloud Platform allocates 2.4 times the requested space to account for snapshot and metadata overhead associated with each LUN. To save disk space, you can set the multiplier to a value >= 1. Setting the multiplier should only be done with extreme care by system administrators who understand the space allocation constraints of the NetApp filer. If you try to set the amount to less than 1, for example, in an attempt to pre-allocate very little space for the LUN, the attempt will most likely fail.

Setting the SR *`other-config:enforce_allocation`* parameter to *`true`* resizes the FlexVols to precisely the amount specified by either the *`multiplier`* value above, or the default 2.4 value.

**Note**

This works on new VDI creation in the selected FlexVol, or on all FlexVols during an SR scan and overrides any manual size adjustments made by the administrator to the SR FlexVols.

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
```

## Managing VDIs in a NetApp SR

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

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

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

## Taking VDI snapshots with a NetApp SR

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 you must snapshot each of the VMs disks in sequence. Because 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 for this value, passed in as an `epochhint`. The first time the `epochhint` value is received, the backend generates a new snapshot based on the cookie name. Any subsequent snapshot requests with the same `epochhint` value will be backed off the existing snapshot:

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

During NetApp SR provisioning, additional disk space is reserved for snapshots. If you plan to not use the snapshotting functionality, you might want to free up this reserved space. To do so, you can reduce the value of the *other-config:multiplier* parameter. By default the value of the multiplier is 2.4, so the amount of space reserved is 2.4 times the amount of space that would be needed for the FlexVols themselves.

## Software iSCSI Support

Xen Cloud Platform provides support for shared SRs on iSCSI LUNs. iSCSI is supported using the open-iSCSI software iSCSI initiator or by using a supported iSCSI Host Bus Adapter (HBA). The steps for using iSCSI HBAs are identical to those for Fibre Channel HBAs, both of which are described in the section called “Creating a shared LVM over Fibre Channel / iSCSI HBA or SAS SR (lvmohba)”.

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 Xen Cloud Platform host in a resource pool and migrated between them with no noticeable downtime.

iSCSI SRs use 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.

### Xen Cloud Platform 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.

Xen Cloud Platform 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 using iSCSI initiator IQN lists, so all iSCSI targets/LUNs to be accessed by a Xen Cloud Platform 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 Xen Cloud Platform hosts in a resource pool, ensure that multi-initiator access is enabled for the specified LUN.

---

The Xen Cloud Platform host IQN value can be adjusted using 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 Xen Cloud Platform host IQN with iSCSI SRs attached. Doing so can result in failures connecting to new targets or existing SRs.

## Managing Hardware Host Bus Adapters (HBAs)

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

### Sample QLogic iSCSI HBA setup

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

Once the HBA is physically installed into the Xen Cloud Platform 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>
   ```

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

### Removing HBA-based SAS, FC or iSCSI device entries

**Note**

This step is not required. Xen.org 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-scsibus` in the format <SCSIid>-<adapter>:<bus>:<target>:<lun> and a standard de-

vice 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 Xen Cloud Platform host database. See the section called “Destroying or forgetting a SR” for details.
2. Remove the zoning configuration within the SAN for the desired LUN to the desired host.
3. Use the **sr-probe** command to determine the ADAPTER, BUS, TARGET, and LUN values corresponding to the LUN to be removed. See the section called “Probing an SR” for details.
4. Remove the device entries with the following command:

```
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.

## LVM over iSCSI

The LVM over iSCSI type represents disks as Logical Volumes within a Volume Group created on an iSCSI LUN.

### 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 that hosts the SR | yes |
| targetIQN | the IQN target address of iSCSI filer that hosts the SR | yes |
| SCSIid | the SCSI bus ID of the destination LUN | yes |
| chapuser | the username to be used for CHAP authentication | no |
| chappassword | the password to be used for CHAP authentication | no |
| port | the network port number on which to query the target | no |

| Parameter Name | Description | Optional? |
|---|---|---|
| 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=<target_ip=> device-config:targetIQN=<target_iqn=> \
device-config:SCSIid=<scsci_id> \
type=lvmoiscsi
```

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

SRs of type lvmohba can be created and managed using the xe CLI.

Device-config parameters for lvmohba SRs:

| Parameter name | Description | Required? |
|---|---|---|
| SCSIid | Device 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 Xen Cloud Platform host in the pool. This process is highly specific to the SAN equipment in use. Please refer to your SAN documentation for details.
2. If necessary, use the HBA CLI included in the Xen Cloud Platform host to configure the HBA:
   - Emulex: `/usr/sbin/hbanyware`
   - QLogic FC: `/opt/QLogic_Corporation/SANsurferCLI`
   - QLogic iSCSI: `/opt/QLogic_Corporation/SANsurferiCLI`

   See the section called “Managing Hardware Host Bus Adapters (HBAs)” for an example of QLogic iSCSI HBA configuration. For more information on Fibre Channel and iSCSI HBAs please refer to the Emulex and QLogic websites.
3. Use the **sr-probe** command to determine the global device path of the HBA LUN. **sr-probe** forces a re-scan of HBAs installed in the system to detect any new LUNs that have been zoned to the host and returns a list of properties for each LUN found. Specify 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:SCSIid=<device_scsi_id> type=lvmohba
```

## NFS VHD

The NFS VHD type stores disks as VHD files on a remote NFS filesystem.

NFS is a ubiquitous form of storage infrastructure that is available in many environments. Xen Cloud Platform 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 Xen Cloud Platform 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 provides a list of valid destination paths exported by the server on which the SR can be created. The NFS server must be configured to export the specified path to all Xen Cloud Platform hosts in the pool, or the creation of the SR and the plugging of the PBD record will fail.

As mentioned at the beginning of this chapter, VDIs stored on NFS are sparse. The image file is allocated as the VM writes data into the disk. This has the considerable benefit that VM image files take up only as much space on the NFS storage 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.

**Note**

The maximum supported length of VHD chains is 30.

As VHD-based images require extra metadata to support sparseness and chaining, the format is not as high-performance as LVM-based storage. In cases where performance really matters, it is well worth forcibly allocating the sparse regions of an image file. This will improve performance at the cost of consuming additional disk space.

Xen Cloud Platform's NFS and VHD implementations 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.

Xen Cloud Platform has been tuned for enterprise-class storage that use non-volatile RAM to provide fast acknowledgments of write requests while maintaining a high degree of data

protection from failure. Xen Cloud Platform has been tested extensively against Network Appliance FAS270c and FAS3020c storage, using Data OnTap 7.2.2.

In situations where Xen Cloud Platform is used with lower-end storage, it will cautiously wait 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 storage to present the SR mount point as an asynchronous mode export. Asynchronous exports acknowledge writes that are not actually on disk, and so administrators should consider the risks of failure carefully in these situations.

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

**Warning**

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

### 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, to the NFS server that hosts the SR | Yes |

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

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

## LVM over hardware HBA

The LVM over hardware HBA type represents disks as VHDs on Logical Volumes within a Volume Group created on an HBA LUN providing, for example, hardware-based iSCSI or FC support.

Xen Cloud Platform 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 Xen Cloud Platform 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 the FC LUN as if it were a locally attached SCSI device.

Use the **sr-probe** command to list the LUN-backed SCSI devices present on the host. This command forces a scan for new LUN-backed SCSI devices. The path value returned by **sr-probe** for a LUN-backed SCSI device is consistent across all hosts with access to the LUN, and therefore must be used when creating shared SRs accessible by all hosts in a resource pool.

The same features apply to QLogic iSCSI HBAs.

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

**Note**

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

## Xen.org StorageLink Gateway (CSLG) SRs

The CSLG storage repository allows use of the Xen.org StorageLink service for native access to a range of iSCSI and Fibre Channel arrays and automated fabric/initiator and array configuration features. Installation and configuration of the StorageLink service is required, for more information please see the StorageLink documentation.

CSLG SRs can be created using the xe CLI only. After creation CSLG SRs can be viewed and managed using the xe CLI.

Because the CSLG SR can be used to access different storage arrays, the exact features available for a given CSLG SR depend on the capabilities of the array. All CSLG SRs use a LUN-per-VDI model where a new LUN is provisioned for each virtual disk. (VDI).

CSLG SRs can co-exist with other SR types on the same storage array hardware, and multiple CSLG SRs can be defined within the same resource pool.

The StorageLink service can be configured using the StorageLink Manager or from within the Xen Cloud Platform control domain using the StorageLink Command Line Interface (CLI). To run the StorageLink (CLI) use the following command, where *<hostname>* is the name or IP address of the machine running the StorageLink service:

```
/opt/Xen.org/StorageLink/bin/csl \
server=<hostname>[:<port>][,<username>,<password>]
```

For more information about the StorageLink CLI please see the StorageLink documentation or use the **/opt/Xen.org/StorageLink/bin/csl help** command.

### Creating a shared StorageLink SR

SRs of type CSLG can only be created by using the xe Command Line Interface (CLI). Once created CSLG SRs can be managed using the xe CLI.

The *device-config* parameters for CSLG SRs are:

| Parameter name | Description | Optional? |
|---|---|---|
| target | The server name or IP address of the machine running the StorageLink service | No |
| storageSystemId | The storage system ID to use for allocating storage | No |
| storagePoolId | The storage pool ID within the specified storage system to use for allocating storage | No |
| username | The username to use for connection to the StorageLink service | Yes * |
| password | The password to use for connecting to the StorageLink service | Yes * |
| cslport | The port to use for connecting to the StorageLink service | Yes * |
| chapuser | The username to use for CHAP authentication | Yes |
| chappassword | The password to use for CHAP authentication | Yes |
| protocol | Specifies the storage protocol to use (fc or iscsi) for multi-protocol storage systems. If not specified fc is used if available, otherwise iscsi. | Yes |
| provision-type | Specifies whether to use thick or thin provisioning (thick or thin); default is thick | Yes |
| provision-options | Additional provisioning options: Set to `dedup` to use the de-duplication features supported by the storage system | Yes |

| Parameter name | Description | Optional? |
|---|---|---|
| raid-type | The level of RAID to use for the SR, as supported by the storage array | Yes |

*If the username, password, or port configuration of the StorageLink service are changed from the default then the appropriate parameter and value must be specified.

SRs of type cslg support two additional parameters that can be used with storage arrays that support LUN grouping features, such as NetApp flexvols.

The `sm-config` parameters for CSLG SRs are:

| Parameter name | Description | Optional? |
|---|---|---|
| pool-count | Creates the specified number of groups on the array, in which LUNs provisioned within the SR will be created | Yes |
| physical-size | The total size of the SR in MB. Each pool will be created with a size equal to physical-size divided by pool-count. | Yes * |

*Required when specifying the `sm-config:pool-count parameter`

---

**Note**

When a new NetApp SR is created using StorageLink, by default a single FlexVol is created for the SR that contains all LUNs created for the SR. To change this behavior and specify the number of FlexVols to create and the size of each FlexVol, use the *`sm-config:pool-size`* and *`sm-config:physical-size`* parameters. The *`sm-config:pool-size`* parameter specifies the number of FlexVols. The *`sm-config:physical-size`* parameter specifies the total size of all FlexVols to be created, so that each FlexVol will be of size *`sm-config:physical-size`* divided by *`sm-config:pool-size`*.

---

### To create a CSLG SR

1. Install the StorageLink service onto a Windows host or virtual machine
2. Configure the StorageLink service with the appropriate storage adapters and credentials
3. Use the **sr-probe** command with the *`device-config:target`* parameter to identify the available storage system IDs

```
xe sr-probe type=cslg device-config:target=192.168.128.10

<csl__storageSystemInfoList>
    <csl__storageSystemInfo>
        <friendlyName>5001-4380-013C-0240</friendlyName>
        <displayName>HP EVA (5001-4380-013C-0240)</displayName>
        <vendor>HP</vendor>
        <model>EVA</model>
        <serialNum>50014380013C0240</serialNum>
        <storageSystemId>HP__EVA__50014380013C0240</storageSystemId>
        <systemCapabilities>
            <capabilities>PROVISIONING</capabilities>
            <capabilities>MAPPING</capabilities>
            <capabilities>MULTIPLE_STORAGE_POOLS</capabilities>
            <capabilities>DIFF_SNAPSHOT</capabilities>
            <capabilities>CLONE</capabilities>
        </systemCapabilities>
        <protocolSupport>
            <capabilities>FC</capabilities>
        </protocolSupport>
        <csl__snapshotMethodInfoList>
            <csl__snapshotMethodInfo>
                <name>5001-4380-013C-0240</name>
                <displayName></displayName>
                <maxSnapshots>16</maxSnapshots>
                <supportedNodeTypes>
                    <nodeType>STORAGE_VOLUME</nodeType>
                </supportedNodeTypes>
                <snapshotTypeList>
                </snapshotTypeList>
                <snapshotCapabilities>
                </snapshotCapabilities>
            </csl__snapshotMethodInfo>
            <csl__snapshotMethodInfo>
                <name>5001-4380-013C-0240</name>
                <displayName></displayName>
                <maxSnapshots>16</maxSnapshots>
                <supportedNodeTypes>
                    <nodeType>STORAGE_VOLUME</nodeType>
                </supportedNodeTypes>
                <snapshotTypeList>
                    <snapshotType>DIFF_SNAPSHOT</snapshotType>
                </snapshotTypeList>
                <snapshotCapabilities>
                </snapshotCapabilities>
            </csl__snapshotMethodInfo>
            <csl__snapshotMethodInfo>
                <name>5001-4380-013C-0240</name>
                <displayName></displayName>
                <maxSnapshots>16</maxSnapshots>
                <supportedNodeTypes>
                    <nodeType>STORAGE_VOLUME</nodeType>
                </supportedNodeTypes>
                <snapshotTypeList>
                    <snapshotType>CLONE</snapshotType>
                </snapshotTypeList>
               <snapshotCapabilities>
               </snapshotCapabilities>
            </csl__snapshotMethodInfo>
        </csl__snapshotMethodInfoList>
    </csl__storageSystemInfo>
</csl__storageSystemInfoList>
```

You can use grep to filter the sr-probe output to just the storage pool IDs

```
xe sr-probe type=cslg device-config:target=192.168.128.10 | grep storageSystemId
        <storageSystemId>EMC__CLARIION__APM00074902515</storageSystemId>
        <storageSystemId>HP__EVA__50014380013C0240</storageSystemId>
        <storageSystemId>NETAPP__LUN__0AD4F00A</storageSystemId>
```

4. Add the desired storage system ID to the **sr-probe** command to identify the storage pools available within the specified storage system

```
xe sr-probe type=cslg \
device-config:target=192.168.128.10  \ device-config:storageSystemId=HP__EVA__50014380013C
<?xml version="1.0" encoding="iso-8859-1"?>
<csl__storagePoolInfoList>
    <csl__storagePoolInfo>
        <displayName>Default Disk Group</displayName>
        <friendlyName>Default Disk Group</friendlyName>
        <storagePoolId>00010710B4080560B6AB0800008000000000000400</storagePoolId>
        <parentStoragePoolId></parentStoragePoolId>
        <storageSystemId>HP__EVA__50014380013C0240</storageSystemId>
        <sizeInMB>1957099</sizeInMB>
        <freeSpaceInMB>1273067</freeSpaceInMB>
        <isDefault>No</isDefault>
        <status>0</status>
        <provisioningOptions>
            <supportedRaidTypes>
                <raidType>RAID0</raidType>
                <raidType>RAID1</raidType>
                <raidType>RAID5</raidType>
            </supportedRaidTypes>
            <supportedNodeTypes>
                <nodeType>STORAGE_VOLUME</nodeType>
            </supportedNodeTypes>
        <supportedProvisioningTypes>
        </supportedProvisioningTypes>
        </provisioningOptions>
    </csl__storagePoolInfo>
</csl__storagePoolInfoList>
```

You can use grep to filter the sr-probe output to just the storage pool IDs

```
xe sr-probe type=cslg \
device-config:target=192.168.128.10 \
device-config:storageSystemId=HP__EVA__50014380013C0240 \
| grep storagePoolId
<storagePoolId>00010710B4080560B6AB0800008000000000000400</storagePoolId>
```

5. Create the SR specifying the desired storage system and storage pool IDs

```
xe sr-create type=cslg name-label=CSLG_EVA_1 shared=true \
device-config:target=192.168.128.10 \
device-config:storageSystemId=HP__EVA__50014380013C0240 \
device-config:storagePoolId=00010710B4080560B6AB0800008000000000000400
```

# Managing Storage Repositories

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

## Destroying or forgetting a SR

You can destroy an SR, which actually deletes the contents of the SR from the physical media. Alternatively you can forget an SR, which allows you to re-attach the SR, for example, to another Xen Cloud Platform host, without removing any of the SR contents. In both cases, the PBD of the SR must first be unplugged.

1. Unplug the PBD to detach the SR from the corresponding Xen Cloud Platform host:

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

2. To destroy the SR, which deletes both the SR and corresponding PBD from the Xen Cloud Platform 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 Xen Cloud Platform host database but leaves the actual SR contents intact on the physical media:

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

---

**Note**

It might take some time for the software object corresponding to the SR to be garbage collected.

---

## Introducing an SR

Introducing an SR that has been forgotten requires introducing an SR, creating a PBD, and manually plugging the PBD to the appropriate Xen Cloud Platform hosts 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 the **sr-probe** command. The UUID of the new SR is 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 is 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.

## Resizing an SR

If you have resized the LUN on which a iSCSI or HBA SR is based, use the following procedures to reflect the size change in Xen Cloud Platform:

1. `iSCSI SRs` - unplug all PBDs on the host that reference LUNs on the same target. This is required to reset the iSCSI connection to the target, which in turn will allow the change in LUN size to be recognized when the PBDs are replugged.
2. `HBA SRs` - reboot the host.

**Note**

In previous versions of Xen Cloud Platform explicit commands were required to resize the physical volume group of iSCSI and HBA SRs. These commands are now issued as part of the PBD plug operation and are no longer required.

## Converting local Fibre Channel SRs to shared SRs

Use the xe CLI to convert a local FC SR to a shared FC SR:

1. Upgrade all hosts in the resource pool to Xen Cloud Platform 0.1.
2. Ensure all hosts in the pool have the SR's LUN zoned appropriately. See the section called “Probing an SR” for details on using the **sr-probe** command 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. Select the SR and then select the **Storage > Repair Storage Repository** menu option.
5. Click **Repair** to create and plug a PBD for each host in the pool.

## Moving Virtual Disk Images (VDIs) between SRs

The set of VDIs associated with a VM can be copied from one SR to another to accommodate maintenance requirements or tiered storage configurations. The xe CLI can be used to copy individual VDIs.

## Adjusting the disk IO scheduler

For general performance, the default disk scheduler `noop` is applied on all new SR types. The `noop` scheduler provides the fairest performance for competing VMs accessing the same device. To apply disk QoS (see the section called “Virtual disk QoS settings”) it is necessary to override the default setting and assign the `cfq` disk scheduler to the SR. The corresponding PBD must be unplugged and re-plugged for the scheduler parameter to take effect. The disk scheduler can be adjusted using the following command:

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

**Note**

This will not effect EqualLogic, NetApp or NFS storage.

# Virtual disk QoS settings

Virtual disks have an optional I/O priority Quality of Service (QoS) setting. This setting can be applied to existing virtual disks using the xe CLI as described in this section.

In the shared SR case, where multiple hosts are accessing the same LUN, the QoS setting is applied to VBDs accessing the LUN from the same host. QoS is not applied across hosts in the pool.

Before configuring any QoS parameters for a VBD, ensure that the disk scheduler for the SR has been set appropriately. See the section called “Adjusting the disk IO scheduler” for 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**

Remember to set the scheduler to `cfq` on the SR, and to ensure 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_param` parameter. For virtual disks, `qos_algorithm_param` takes a sched key, and depending on the value, also requires a class key.

Possible values of `qos_algorithm_param: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 7 is the highest priority and 0 is the lowest, so that, for example, I/O requests with a priority of 5, will be given priority over I/O requests with a priority of 2.

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 sr-param-set uuid=<sr_uuid> other-config:scheduler-cfq
xe pbd-plug uuid=<pbd_uuid>
```

# Chapter 4. Networking

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

**Note**

Xen Cloud Platform provides automated configuration and management of NICs using the xe command line interface (CLI). Unlike previous Xen Cloud Platform versions, the host networking configuration files should not be edited directly in most cases; where a CLI command is available, do not edit the underlying files.

If you are already familiar with Xen Cloud Platform networking concepts, you may want to skip ahead to one of the following sections:

- For procedures on how to create networks for standalone Xen Cloud Platform hosts, see the section called “Creating networks in a standalone server”.
- For procedures on how to create networks for Xen Cloud Platform hosts that are configured in a resource pool, see the section called “Creating networks in resource pools”.
- For procedures on how to create VLANs for Xen Cloud Platform hosts, either standalone or part of a resource pool, see the section called “Creating VLANs”.
- For procedures on how to create bonds for standalone Xen Cloud Platform hosts, see the section called “Creating NIC bonds on a standalone host”.
- For procedures on how to create bonds for Xen Cloud Platform hosts that are configured in a resource pool, see the section called “Creating NIC bonds in resource pools”.

## Xen Cloud Platform networking overview

This section describes the general concepts of networking in the Xen Cloud Platform environment.

**Note**

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

### Network objects

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

- A *PIF*, which represents a physical network interface on a Xen Cloud Platform host. PIF objects have a name and description, a globally unique UUID, the parameters of the NIC that they represent, and the network and server they are connected to.

- A *VIF*, which represents a virtual interface on a Virtual Machine. VIF objects have a name and description, a globally unique UUID, and the network and VM they are connected to.
- A *network*, which is a virtual Ethernet switch on a Xen Cloud Platform host. Network objects have a name and description, a globally unique UUID, and the collection of VIFs and PIFs connected to them.

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

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

## Networks

Each Xen Cloud Platform 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 Xen Cloud Platform 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.

## VLANs

Virtual Local Area Networks (VLANs), as defined by the IEEE 802.1Q standard, allow a single physical network to support multiple logical networks. XenServer hosts can work with VLANs in multiple ways.

---

**Note**

All supported VLAN configurations are equally applicable to pools and standalone hosts, and bonded and non-bonded configurations.

---

### Using VLANs with host management interfaces

Switch ports configured to perform 802.1Q VLAN tagging/untagging, commonly referred to as ports with a *native VLAN* or as *access mode* ports, can be used with XenServer management interfaces to place management traffic on a desired VLAN. In this case the XenServer host is unaware of any VLAN configuration.

XenServer management interfaces cannot be assigned to a XenServer VLAN via a trunk port.

### Using VLANs with virtual machines

Switch ports configured as 802.1Q VLAN trunk ports can be used in combination with the XenServer VLAN features to connect guest virtual network interfaces (VIFs) to specific VLANs. In this case the XenServer host performs the VLAN tagging/untagging functions for the guest, which is unaware of any VLAN configuration.

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

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

### Using VLANs with dedicated storage NICs

Dedicated storage NICs can be configured to use native VLAN / access mode ports as described above for management interfaces, or with trunk ports and XenServer VLANs as described above for virtual machines. To configure dedicated storage NICs, see the section called “Configuring a dedicated storage NIC”.

### Combining management interfaces and guest VLANs on a single host NIC

A single switch port can be configured with both trunk and native VLANs, allowing one host NIC to be used for a management interface (on the native VLAN) and for connecting guest VIFs to specific VLAN IDs.

## NIC bonds

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

Xen Cloud Platform 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.

Xen Cloud Platform NIC bonds are represented by additional PIFs. The bond PIF can then be connected to a Xen Cloud Platform 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.

Xen Cloud Platform supports Source Level Balancing (SLB) NIC bonding. SLB bonding:

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

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

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

Xen Cloud Platform bonded PIFs do not require IP configuration for the bond when used for guest traffic. This is because the bond operates at Layer 2 of the OSI, the data link layer, and no IP addressing is used at this layer. When used for non-guest traffic (to connect to it with XenCenter for management, or to connect to shared network storage), one IP configuration is required per bond. (Incidentally, this is true of unbonded PIFs as well, and is unchanged from Xen Cloud Platform 4.1.0.)

Gratuitous ARP packets are sent when assignment of traffic changes from one interface to another as a result of fail-over.

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

---

**Note**

Bonding is set up with an Up Delay of 31000ms and a Down Delay of 200ms. The seemingly long Up Delay is purposeful because of the time taken by some switches to actually start routing traffic. Without it, when a link comes back after failing, the bond might rebalance traffic onto it before the switch is ready to pass traffic. If you want to move both connections to a different switch, move one, then wait 31 seconds for it to be used again before moving the other.

---

## Initial networking configuration

The Xen Cloud Platform host networking configuration 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 Xen Cloud Platform host has a single NIC, the follow configuration is 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 Xen Cloud Platform 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 using the network object which acts as a virtual Ethernet switch.

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

# Managing networking configuration

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

## Creating networks in a standalone server

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

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

### To add a new network using the CLI

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

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

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

## Creating networks in resource pools

All Xen Cloud Platform 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 Xen Cloud Platform host is joined to a pool.

Having the same physical networking configuration for Xen Cloud Platform hosts within a pool is important because all hosts in a pool share a common set of Xen Cloud Platform networks. PIFs on the individual hosts are connected to pool-wide networks based on device name. For example, all Xen Cloud Platform 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 Xen Cloud Platform host in the pool.

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

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

## Creating VLANs

For servers in a resource pool, you can use the **pool-vlan-create** command. This command creates the VLAN and automatically creates and plugs in the required PIFs on the hosts in the pool. See the section called “pool-vlan-create” for more information.

### To connect a network to an external VLAN using the CLI

1. Open the Xen Cloud Platform 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 on all VMs to be connected to the new VLAN. A new PIF will be created and plugged into the specified network. The UUID of the new PIF object is returned.

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

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

# Creating NIC bonds on a standalone host

This section describes how to use the xe CLI to create bonded NIC interfaces on a standalone Xen Cloud Platform host. See the section called “Creating NIC bonds in resource

pools" for details on using the xe CLI to create NIC bonds on Xen Cloud Platform hosts that comprise a resource pool.

## Creating a NIC bond on a dual-NIC host

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

### Bonding two NICs together

1. Use 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; separated by commas, specify the newly created network UUID and the UUIDs of the PIFs to be bonded. The UUID for the bond is returned:

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

> **Note**
>
> See the section called "Controlling the MAC address of the bond" for details on controlling the MAC address used for the bond PIF.

5. Use the **pif-list** command to determine the UUID of the new bond PIF:

```
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 8, *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_management_pif_uuid> mode=None
```

9. Move existing VMs to the bond network using the **vif-destroy** and **vif-create** commands.
10. Restart the VMs shut down in step 1.

## Controlling the MAC address of the bond

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

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

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

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

## Reverting NIC bonds

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

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

# Creating NIC bonds in resource pools

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

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

**Warning**

Do not attempt to create network bonds while HA is enabled. The process of bond creation will disturb the in-progress HA heartbeating and cause hosts to self-fence (shut themselves down); subsequently they will likely fail to reboot properly and will need the **host-emergency-ha-disable** command to recover.

## Adding NIC bonds to new resource pools

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

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

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

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

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

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

   ```
   xe pif-list
   ```

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

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

   **Note**

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

   d. Use the **pif-list** command to determine the UUID of the new bond PIF:

   ```
   xe pif-list  network-uuid=<network_uuid>
   ```

   e. Use the **pif-reconfigure-ip** command to configure the desired management interface IP address settings for the bond PIF. See Chapter 8, *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
   ```

f. 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>
```

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

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

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

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

The network and bond information is automatically replicated to the new host. However, the management interface is not automatically moved from the host NIC to the bonded NIC. Move the management interface on the host to enable the bond as follows:

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

```
xe host-list
```

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

```
xe pif-list network-name-label=<network_name> host-uuid=<host_uuid>
```

c. Use the **pif-reconfigure-ip** command to configure the desired management interface IP address settings for the bond PIF. See Chapter 8, *Command line interface*, for more detail on the options available for the **pif-reconfigure-ip** command. *This command must be run directly on the host:*

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

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

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

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

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

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

## Adding NIC bonds to an existing pool

### Warning

Do not attempt to create network bonds while HA is enabled. The process of bond creation disturbs the in-progress HA heartbeating and causes hosts to self-fence (shut themselves down); subsequently they will likely fail to reboot properly and you will need to run the **host-emergency-ha-disable** command to recover them.

### Note

The quickest way to create pool-wide NIC bonds is to create the bond on the master, and then restart the other pool members. Alternately you can use the **service xapi restart** command. This causes the bond and VLAN settings on the master to be inherited by each host. The management interface of each host must, however, be manually reconfigured.

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 other hosts 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 other hosts.
3. The **bond-create**, **host-management-reconfigure** and **host-management-disable** commands affect the host on which they are run and so are not suitable for use on one host in a pool to change the configuration of another. Run these commands directly on the console of the host to be affected.

### To add NIC bonds to the pool master and other 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 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 the section called "Controlling the MAC address of the bond" for details on controlling the MAC address used for the bond PIF.

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

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

```
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 host*:

```
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 host*:

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

10. Move existing VMs to the bond network using the **vif-destroy** and **vif-create** commands.
11. Repeat steps 3 - 10 for other hosts.
12. Restart the VMs previously shut down.

## Configuring a dedicated storage NIC

Xen Cloud Platform uses the xe CLI to configure and dedicate a NIC to specific functions, such as storage traffic.

Assigning a NIC to a specific function will prevent the use of the NIC for other functions such as host management, but requires that the appropriate network configuration be in place in order to ensure the NIC is used for the desired traffic. For example, to dedicate a NIC to storage traffic the NIC, storage target, switch, and/or VLAN must be configured such

that the target is only accessible over the assigned NIC. This allows use of standard IP routing to control how traffic is routed between multiple NICs within a Xen Cloud Platform.

**Note**

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

### To assign NIC functions using the xe CLI

1. Ensure that the PIF is on a separate subnet, or routing is configured to suit your network topology in order to force the desired traffic over the selected PIF.
2. Setup an IP configuration for the PIF, adding appropriate values for the mode parameter and if using static IP addressing the IP, netmask, gateway, and DNS parameters:

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

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

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

```
xe pif-param-set other-config:management_purpose="Storage" uuid=<pif-uuid>
```

If you want to use a storage interface that can be routed from the management interface also (bearing in mind that this configuration is not recommended), then you have two options:

- After a host reboot, ensure that the storage interface is correctly configured, and use the **xe pbd-unplug** and **xe pbd-plug** commands to reinitialize the storage connections on the host. This will restart the storage connection and route it over the correct interface.
- Alternatively, you can use **xe pif-forget** to remove the interface from the Xen Cloud Platform database, and manually configure it in the control domain. This is an advanced option and requires you to be familiar with how to manually configure Linux networking.

## Controlling Quality of Service (QoS)

Xen.org Essentials for Xen Cloud Platform allows an optional Quality of Service (QoS) value to be set on VM virtual network interfaces (VIFs) using the CLI. The supported QoS algorithm type is rate limiting, specified as a maximum transfer rate for the VIF in Kb per second.

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
```

# Changing networking configuration options

This section discusses how to change the networking configuration of a Xen Cloud Platform host. This includes:

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

## Hostname

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

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

The underlying control domain hostname changes dynamically to reflect the new hostname.

## DNS servers

To add or remove DNS servers in the IP addressing configuration of a Xen Cloud Platform 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>
```

## Changing IP address configuration for a standalone host

Network interface configuration can be changed using the xe CLI. The underlying network configuration scripts should not be modified directly.

To modify the IP address configuration of a PIF, use the **pif-reconfigure-ip** CLI command. See the section called “pif-reconfigure-ip” for details on the parameters of the **pif-reconfigure-ip** command.

**Note**

See the section called “Changing IP address configuration in resource pools” for details on changing host IP addresses in resource pools.

## Changing IP address configuration in resource pools

Xen Cloud Platform 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 other hosts.

**Note**

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

### Changing the IP address of a pool member host

1. Use the **pif-reconfigure-ip** CLI command to set the IP address as desired. See Chapter 8, *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 Xen Cloud Platform hosts in the pool are visible:

```
xe host-list
```

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

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

### To change the IP address of a pool master host

1. Use the **pif-reconfigure-ip** CLI command to set the IP address as desired. See Chapter 8, *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 Xen Cloud Platform host, use the **pool-recover-slaves** command to force the master to contact each of the member hosts and inform them of the new master IP address:

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

Refer to the section called “Master failures” for more information on emergency mode.

## Management interface

When Xen Cloud Platform is installed on a host with multiple NICs, one NIC is selected for use as the management interface.

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

1. Use the **pif-list** command to determine which PIF corresponds to the NIC to be used as the management interface. The UUID of each PIF is 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 8, *Command line interface* for more detail on the options available for the **pif-reconfigure-ip** command.

```
xe pif-param-list uuid=<pif_uuid>
```

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

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

**Warning**

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

## Disabling management access

To disable remote access to the management console entirely, use the **host-management-disable** CLI command.

## Adding a new physical NIC

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

# NIC/PIF ordering in resource pools

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

## Verifying NIC ordering

Use the **pif-list** command to verify that NIC ordering is consistent across your Xen Cloud Platform 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.

## 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 Xen Cloud Platform host must be standalone and not joined to a resource pool.
- Re-ordering a PIF configured as the management interface of the host 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 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 8, *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>
```

# Networking Troubleshooting

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

## Diagnosing network corruption

Some network card models 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.

---

**Warning**

Disabling receive / transmit offload optimizations 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.

## 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 Xen Cloud Platform host.

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

- Xen.org recommends that you ensure networking configuration is set up correctly before creating a resource pool, as it is usually easier to recover from a bad configuration in a non-pooled state.
- The **host-management-reconfigure** and **host-management-disable** commands affect the Xen Cloud Platform 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 Xen Cloud Platform 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. Workload Balancing

## Workload Balancing Overview

Workload Balancing is a Xen Cloud Platform feature that helps you balance virtual machine workloads across hosts and locate VMs on the best possible servers for their workload in a resource pool. When Workload Balancing places a virtual machine, it determines the best host on which to start a virtual machine or it rebalances the workload across hosts in a pool. For example, Workload Balancing lets you determine where to:

- Start a virtual machine
- Resume a virtual machine that you powered off
- Move virtual machines when a host fails

When Workload Balancing is enabled, if you put a host into Maintenance Mode, Workload Balancing selects the optimal server for each of the host's virtual machines. For virtual machines taken offline, Workload Balancing provides recommendations to help you restart virtual machines on the optimal server in the pool.

Workload Balancing also lets you balance virtual-machine workloads across hosts in a Xen Cloud Platform resource pool. When the workload on a host exceeds the level you set as acceptable (the threshold), Workload Balancing will make recommendations to move part of its workload (for example, one or two virtual machines) to a less-taxed host in the same pool. It does this by evaluating the existing workloads on hosts against resource performance on other hosts.

You can also use Workload Balancing to help determine if you can power off hosts at certain times of day.

Workload Balancing performs these tasks by analyzing Xen Cloud Platform resource-pool metrics and recommending optimizations. You decide if you want these recommendations geared towards resource performance or hardware density. You can fine-tune the weighting of individual resource metrics (CPU, network, memory, and disk) so that the placement recommendations and critical thresholds align with your environment's needs.

To help you perform capacity planning, Workload Balancing provides historical reports about host and pool health, optimization and virtual-machine performance, and virtual-machine motion history.

## Workload Balancing Basic Concepts

Workload Balancing captures data for resource performance on virtual machines and physical hosts. It uses this data, combined with the preferences you set, to provide optimization and placement recommendations. Workload Balancing stores performance data in a SQL Server database: the longer Workload Balancing runs the more precise its recommendations become.

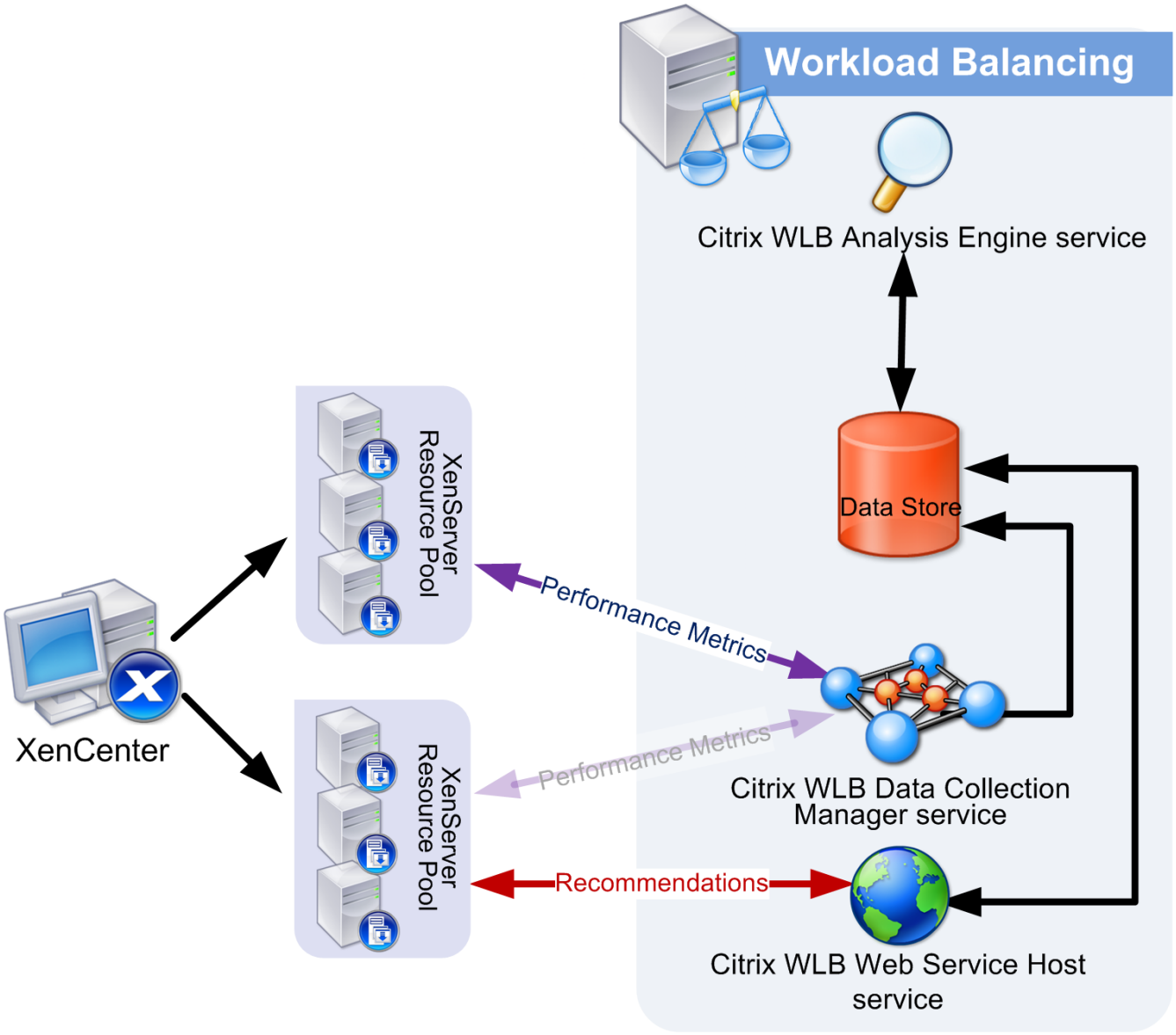
Workload Balancing recommends moving virtual-machine workloads across a pool to get the maximum efficiency, which means either *performance* or *density* depending on your goals. Within a Workload Balancing context:

- **Performance** refers to the usage of physical resources on a host (for example, the CPU, memory, network, and disk utilization on a host). When you set Workload Balancing to maximize performance, it recommends placing virtual machines to ensure the maximum amount of resources are available for each virtual machine.
- **Density** refers to the number of virtual machines on a host. When you set Workload Balancing to maximize density, it recommends placing virtual machines to ensure they have adequate computing power so you can reduce the number of hosts powered on in a pool.

Workload Balancing configuration preferences include settings for placement (performance or density), virtual CPUs, and performance thresholds.

## Workload Balancing Component Overview

The Workload Balancing software is a collection of services and components that let you manage all of basic functions of Workload Balancing, such as managing workload and displaying reports. You can install the Workload Balancing services on one computer (physical or virtual) or multiple computers. A Workload Balancing server can manage more than one resource pool.




Workload Balancing consists of the following components:

- *Workload Balancing server*. Collects data from the virtual machines and their hosts and writes the data to the data store. This service is also referred to as the "data collector."
- *Data Store*. A Microsoft SQL Server or SQL Server Express database that stores performance and configuration data.

For more information about Workload Balancing components for large deployments with multiple servers, see the section called “Multiple Server Deployments”.

## Designing Your Workload Balancing Deployment

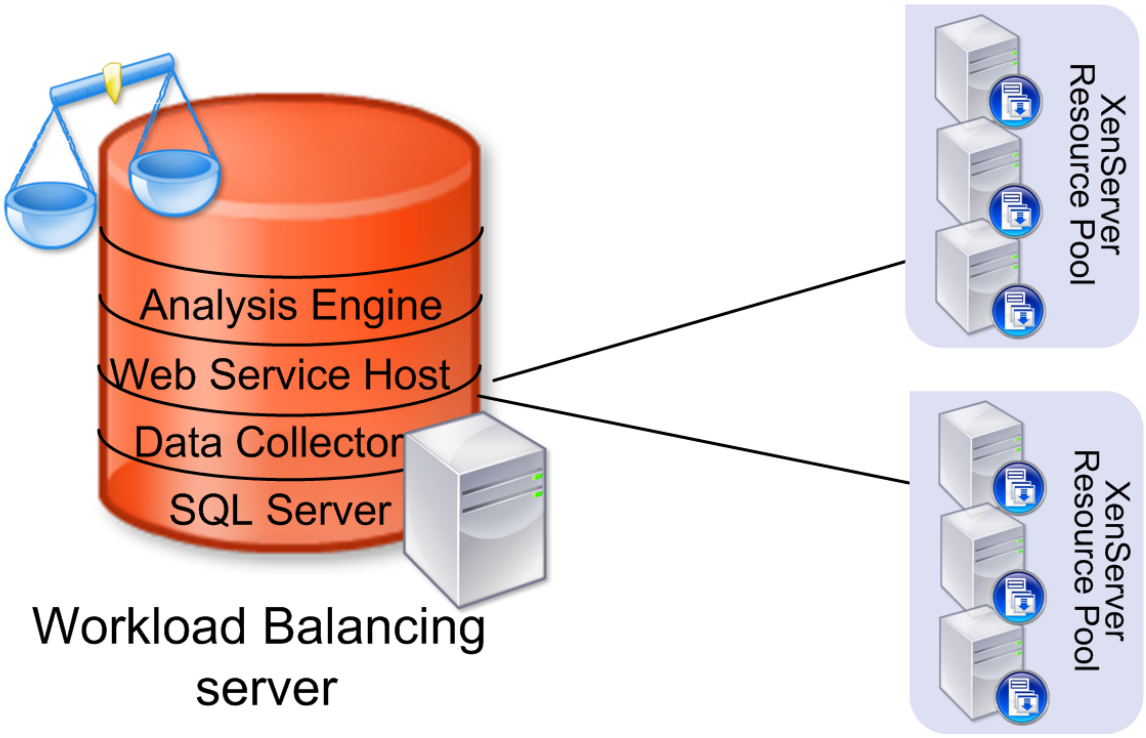
You can install Workload Balancing on one computer (physical or virtual) or distribute the components across multiple computers. The three most common deployment configurations are the following:

- All components are installed on a single server
- The data collector is installed on a dedicated server
- All components are installed on a single server, but the data store is installed on central database server

Because one data collector can monitor multiple resource pools, you do not need multiple data collectors to monitor multiple pools.

### Deploying One Server

Depending on your environment and goals, you can install Workload Balancing and the data store on one server. In this configuration, one data collector monitors all the resource pools.




The following table shows the advantages and disadvantages to a single-server deployment:

| Advantages | Disadvantages |
| --- | --- |
| • Simple installation and configuration.<br>• No Windows domain requirement. | • Single point of failure. |

## Planning for Future Growth

If you anticipate that you will want to add more resource pools in the future, consider designing your Workload Balancing deployment so that it supports growth and scalability. Consider:

- **Using SQL Server for the data store**. In large environments, consider using SQL Server for the data store instead of SQL Server Express. Because SQL Server Express has a 4GB disk-space limit, Workload Balancing limits the data store to 3.5GB when installed on this database. SQL Server has no preset disk-space limitation.
- **Deploying the data store on a dedicated server**. If you deploy SQL Server on a dedicated server (instead of collocating it on the same computer as the other Workload Balancing services), you can let it use more memory.

## Multiple Server Deployments

In some situations, you might need to deploy Workload Balancing on multiple servers. When you deploy Workload Balancing on multiple servers, you place its key services on one more servers:

- *Data Collection Manager service*. Collects data from the virtual machines and their hosts and writes the data to the data store. This service is also referred to as the "data collector."
- *Web Service Host*. Facilitates communications between the Xen Cloud Platform and the Analysis Engine. Requires a security certificate, which you can create or provide during Setup.
- *Analysis Engine service*. Monitors resource pools and determines if a resource pool needs optimizations.

The size of your Xen Cloud Platform environment affects your Workload Balancing design. Since every environment is different, the size definitions that follow are examples of environments of that size:

| Size | Example |
| --- | --- |
| Small | One resource pool with 2 hosts and 8 virtual machines |
| Medium | Two resource pools with 6 hosts and 8 virtual machines per pool |
| Large | Five resource pools with 16 hosts and 64 virtual machines per pool |

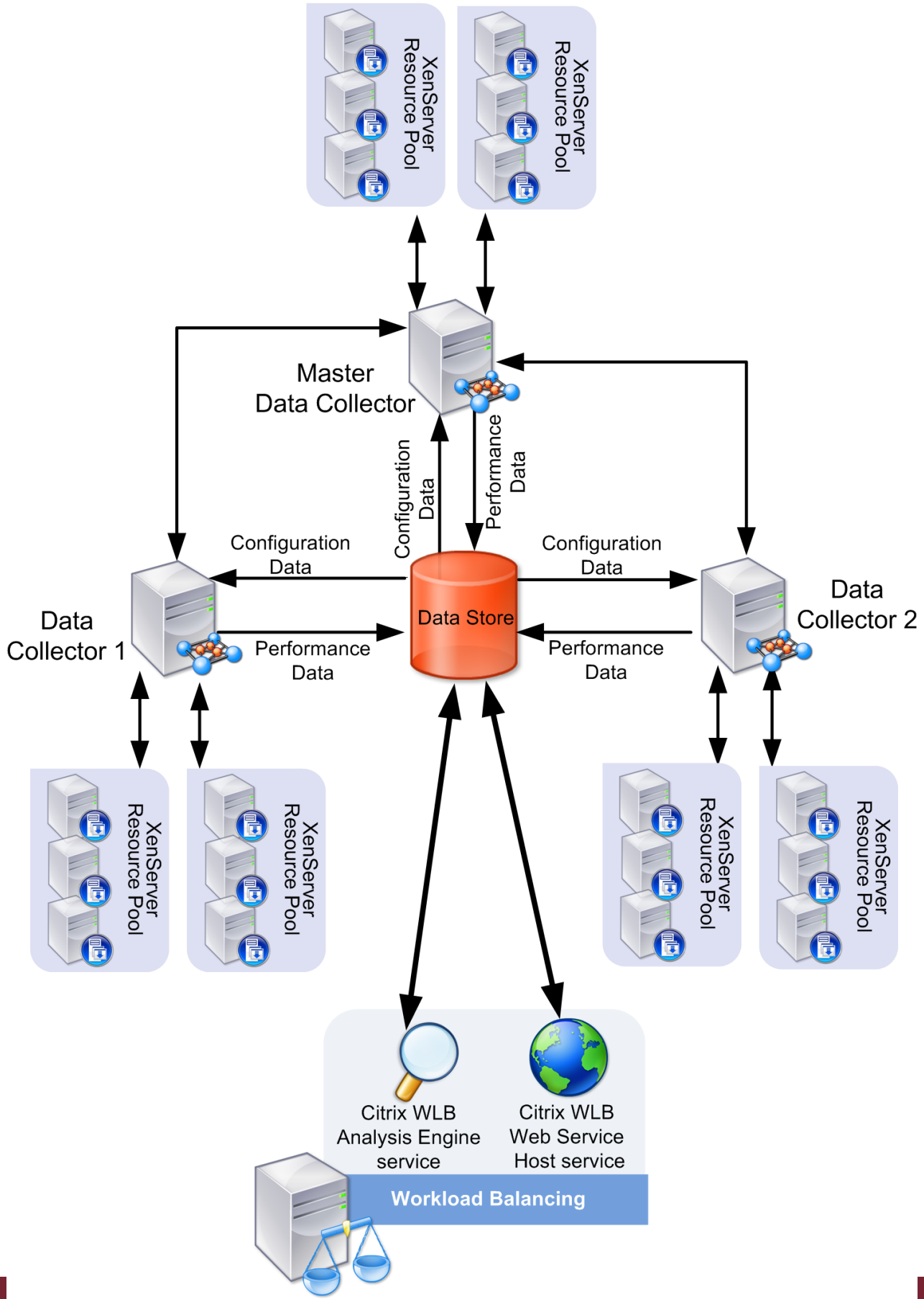
## Deploying Multiple Servers

Having multiple servers for Workload Balancing's services may be necessary in large environments. For example, having multiple servers may reduce "bottlenecks." If you decide to deploy Workload Balancing's services on multiple computers, all servers must be members of mutually trusted Active Directory domains.

| Advantages | Disadvantages |
| --- | --- |
| • Provides better scalability.<br>• Can monitor more resource pools. | • More equipment to manage and, consequently, more expense. |

## Deploying Multiple Data Collectors

Workload Balancing supports multiple data collectors, which might be beneficial in environments with many resource pools. When you deploy multiple data collectors, the data collectors work together to ensure all Xen Cloud Platform pools are being monitored at all times.


XenServer Resource Pool
XenServer Resource Pool
Master Data Collector
Configuration Data
Performance Data
Configuration Data
Configuration Data
Data Collector 1
Data Store
Data Collector 2
Performance Data
Performance Data
XenServer Resource Pool
XenServer Resource Pool
XenServer Resource Pool
XenServer Resource Pool
Citrix WLB Analysis Engine service
Citrix WLB Web Service Host service
Workload Balancing

All data collectors collect data from their own resource pools. One data collector, referred to as the master, also does the following:

- Checks for configuration changes and determines the relationships between resource pools and data collectors
- Checks for new Xen Cloud Platform resource pools to monitor and assigns these pools to a data collector
- Monitors the health of the other data collectors

If a data collector goes offline or you add a new resource pool, the master data collector rebalances the workload across the data collectors. If the master data collector goes offline, another data collector assumes the role of the master.

### Considering Large Environments

In large environments, consider the following:

- When you install Workload Balancing on SQL Server Express, Workload Balancing limits the size of the metrics data to 3.5GB. If the data grows beyond this size, Workload Balancing starts grooming the data, deleting older data, automatically.
- Xen.org recommends putting the data store on one computer and the Workload Balancing services on another computer.
- For Workload Balancing data-store operations, memory utilization is the largest consideration.

## Workload Balancing Security

Xen.org designed Workload Balancing to operate in a variety of environments, and Xen.org recommends properly securing the installation. The steps required vary according to your planned deployment and your organization's security policies. This topic provides information about the available options and makes recommendations.

**Important**

Xen.org does not recommend changing the privileges or accounts under which the Workload Balancing services run.

### Encryption Requirements

Xen Cloud Platform communicates with Workload Balancing using HTTPS. Consequently, you must create or install an SSL/TLS certificate when you install Workload Balancing (or the Web Services Host, if it is on a separate server). You can either use a certificate from a Trusted Authority or create a self-signed certificate using Workload Balancing Setup.

The self-signed certificate Workload Balancing Setup creates is not from a Trusted Authority. If you do not want to use this self-signed certificate, before you begin the setup process, prepare a certificate and specify that certificate when prompted.

If desired, during Workload Balancing Setup, you can export the certificate so that you can import it into Xen Cloud Platform after Setup.

**Note**

If you create a self-signed certificate during Workload Balancing Setup, Xen.org recommends that you eventually replace this certificate with one from a Trusted Authority.

### Domain Considerations

When deploying Workload Balancing, your environment determines your domain and security requirements.

- If your Workload Balancing services are on multiple computers, the computers must be part of a domain.
- If your Workload Balancing components are in separate domains, you must configure trust relationships between those domains.

### SQL Server Authentication Requirements

When you install SQL Server or SQL Server Express, you must configure Windows authentication (also known as Integrated Windows Authentication). Workload Balancing does not support SQL Server Authentication.

## Workload Balancing Installation Overview

Workload Balancing is a Xen Cloud Platform feature that helps manage virtual-machine workloads within a Xen Cloud Platform environment. Workload Balancing requires that you:

1. Install SQL Server or SQL Server Express.
2. Install Workload Balancing on one or more computers (physical or virtual). See the section called “Designing Your Workload Balancing Deployment”.

Typically, you install and configure Workload Balancing after you have created one or more Xen Cloud Platform resource pools in your environment.

You install all Workload Balancing functions, such as the Workload Balancing data store, the Analysis Engine, and the Web Service Host, from Setup.

You can install Workload Balancing in one of two ways:

- **Installation Wizard**. Start the installation wizard from Setup.exe. Xen.org suggests installing Workload Balancing from the installation wizard because this method checks your system meets the installation requirements.
- **Command Line**. If you install Workload Balancing from the command line, the prerequisites are not checked. For Msiexec properties, see the section called “Windows Installer Commands for Workload Balancing”.

When you install the Workload Balancing data store, Setup creates the database. You do not need to run Workload Balancing Setup locally on the database server: Setup supports installing the data store across a network.

If you are installing Workload Balancing services as components on separate computers, you must install the database component before the Workload Balancing services.

After installation, you must configure Workload Balancing before you can use it to optimize workloads. For information, see the section called “Initializing and Configuring Workload Balancing”.

For information about System Requirements, see the section called “Workload Balancing System Requirements”. For installation instructions, see the section called “Installing Workload Balancing”.

# Workload Balancing System Requirements

This topic provides system requirements for:

- the section called “Supported Xen Cloud Platform Versions”
- the section called “Supported Operating Systems”
- the section called “Recommended Hardware”
- the section called “Data Collection Manager”
- the section called “Analysis Engine”
- the section called “Web Service Host”

For information about data store requirements, see the section called “Workload Balancing Data Store Requirements”.

## Supported Xen Cloud Platform Versions

- Xen Cloud Platform 1.0

## Supported Operating Systems

Unless otherwise noted, Workload Balancing components run on the following operating systems (32-bit and 64-bit):

- Windows Server 2008
- Windows Server 2003, Service Pack 2
- Windows Vista
- Windows XP Professional, Service Pack 2 or Service Pack 3

If you are installing with the User Account Control (UAC) enabled, see Microsoft's documentation.

## Recommended Hardware

Unless otherwise noted, Workload Balancing components require the following hardware (32-bit and 64-bit):

- CPU: 2GHz or faster
- Memory: 2GB recommended (1GB of RAM required)
- Disk Space: 20GB (minimum)

When all Workload Balancing services are installed on the same server, Xen.org recommends that the server have a minimum of a dual-core processor.

### Data Collection Manager

| | |
|---|---|
| Operating System Components | • Microsoft .NET Framework 3.5, Service Pack 1 or higher |
| Hard Drive | 1GB |

### Analysis Engine

| | |
|---|---|
| Operating System Components | • Microsoft .NET Framework 3.5, Service Pack 1 or higher |

### Web Service Host

| | | |
|---|---|---|
| Operating System Components | • Microsoft .NET Framework 3.5, Service Pack 1 or higher | |

## Workload Balancing Data Store Requirements

This topic provides information about the SQL Server versions and configurations that Workload Balancing supports. It also provides information about additional compatibility and authentication requirements.

### Installation Requirements for SQL Server

In addition to the prerequisites SQL Server and SQL Server Express require, the data store also requires the following:

**Note**

In this topic, the term *SQL Server* refers to both SQL Server and SQL Server Express unless the version is mentioned explicitly.

| | |
|---|---|
| Operating System | One of the following, as required by your SQL Server edition:<br>• Windows Server 2008<br>• Windows Server 2003, Service Pack 1 or higher<br>• Windows Vista and Windows XP Professional (for SQL Server Express) |
| Database | The 32-bit or 64-bit edition of: |

| | |
|---|---|
| | • SQL Server 2008 Express. The 32-bit edition is available on the Workload Balancing installation media in the **sql** folder.<br>• SQL Server 2008 (Standard edition or better)<br>• SQL Server 2005, Service Pack 1 or higher (Standard edition or better)<br>**Note**<br>Windows Server 2008 servers require SQL Server 2005, Service Pack 2 or higher. |
| Required Configurations | • Configure SQL Server for case-insensitive collation. Workload Balancing does not currently support case-sensitive collation.<br>• Microsoft SQL Server 2005 Backward Compatibility Components. See the section called "Backwards Compatibility Requirement for SQL Server 2008" for more information. |
| Hard Drive | SQL Server Express: 5GB<br>SQL Server: 20GB |

## SQL Server Database Authentication Requirements

During installation, Setup must connect and authenticate to the database server to create the data store. Configure the SQL Server database instance to use either:

- **Windows Authentication mode**, or
- **SQL Server and Windows Authentication mode** (Mixed Mode authentication)

If you create an account on the database for use during Setup, the account must have **sysadmin** privileges for the database instance where you want to create the Workload Balancing data store.

## Backwards Compatibility Requirement for SQL Server 2008

After installing SQL Server Express 2008 or SQL Server 2008, you must install the SQL Server 2005 Backward Compatibility Components on all Workload Balancing computers before running Workload Balancing Setup. The Backward Compatibility components let Workload Balancing Setup configure the database.

The Workload Balancing installation media includes the 32-bit editions of SQL Server Express 2008 and the SQL Server 2005 Backward Compatibility Components.

- While some SQL Server editions may include the Backward Compatibility components with their installation programs, their Setup program might not install them by default.
- You can also obtain the Backward Compatibility components from the download page for the latest Microsoft SQL Server 2008 Feature Pack.

Install the files in the **sql** folder in the following order:

1. **en_sql_server_2008_express_with_tools_x86.exe**. Installs SQL Server Express, 32-bit edition. Requires installing Microsoft .NET Framework 3.5, Service Pack 1 first.
2. **SQLServer2005_BC.msi**. Installs the SQL Server 2005 Backward Compatibility Components for 32-bit computers.

## Operating System Language Support

Workload Balancing is supported on the following operating system languages:

- US English
- Japanese (Native JP)

---

**Note**

In configurations where the database and Web server are installed on separate servers, the operating system languages must match on both computers.

---

## Preinstallation Considerations

You may need to configure software in your environment so that Workload Balancing can function correctly. Review the following considerations and determine if they apply to your environment. Also, check the Xen Cloud Platform readme for additional, late-breaking release-specific requirements.

- **Account for Workload Balancing**. Before Setup, you must create a user account for Xen Cloud Platform to use to connect to Workload Balancing (specifically the Web Service Host service).

  This user account can be either a domain account or an account local to the computer running Workload Balancing (or the Web Service Host service).

  ---

  **Important**

  When you create this account in Windows, Xen.org suggests enabling the **Password never expires** option.

  ---

  During Setup, you must specify the authorization type (a single user or group) and the user or group with permissions to make requests of the Web Service Host service. For additional information, see the section called “Authorization for Workload Balancing ”.
- **SSL/TLS Certificate**. Xen Cloud Platform and Workload Balancing communicate over HTTPS. Consequently, during Workload Balancing Setup, you must provide either an SSL/TLS certificate from a Trusted Authority or create a self-signed certificate.

- **Group Policy**. If the server on which you are installing Workload Balancing is a member of a Group Policy Organizational Unit, ensure that current or scheduled, future policies do not prohibit Workload Balancing or its services from running.

---

**Note**

In addition, review the applicable release notes for release-specific configuration information.

---

## Installing Workload Balancing

Before installing Workload Balancing, you must:

1. Install a SQL Server or SQL Server Express database as described in Workload Balancing Data Store Requirements.
2. Have a login on the SQL Server database instance that has SQL Login creation privileges. For SQL Server Authentication, the account needs **sysadmin** privileges.
3. Create an account for Workload Balancing, as described in Preinstallation Considerations and have its name on hand.
4. Configure all Workload Balancing servers to meet the system requirements described in the section called “Workload Balancing System Requirements”.

After Setup is finished installing Workload Balancing, verify that it is installed correctly before it begins gathering data and making recommendations.

## To install Workload Balancing on a single server

The following procedure installs Workload Balancing and all of its services on one computer:

1. Launch the **Workload Balancing Setup** wizard from Autorun.exe, and select the **Workload Balancing installation** option.
2. After the initial Welcome page appears, click **Next.**
3. In the Setup Type page, select **Workload Balancing Services and Data Store**, and click **Next**. This option lets you install Workload Balancing, including the Web Services Host, Analysis Engine, and Data Collection Manager services. After you click Next, Workload Balancing Setup verifies that your system has the correct prerequisites.
4. Accept the End-User License Agreement.
5. In the Component Selection page, select all of the following components:
   - **Database**. Creates and configures a database for the Workload Balancing data store.
   - **Services** .
   - **Data Collection Manager**. Installs the Data Collection Manager service, which collects data from the virtual machines and their hosts and writes this data to the data store.
   - **Analysis Engine**. Installs the Analysis Engine service, which monitors resource pools and recommends optimizations by evaluating the performance metrics the data collector gathered.

- **Web Service Host**. Installs the service for the Web Service Host, which facilitates communications between Xen Cloud Platform and the Analysis Engine.

  If you enable the Web Service Host component, Setup prompts you for a security certificate. You can either use the self-signed certificate Workload Balancing Setup provides or specify a certificate from a Trusted Authority.

6. In the **Database Server** page, in the **SQL Server Selection** section, select one of the following:
   - **Enter the name of a database server** . Lets you type the name of the database server that will host the data store. Use this option to specify an instance name.

     **Note**

     If you installed SQL Express and specified an instance name, append the server name with **`\yourinstancename`**. If you installed SQL Express without specifying an instance name, append the server name with **`\sqlexpress`**.

   - **Choose an existing database server** . Lets you select the database server from a list of servers Workload Balancing Setup detected on your network. Use the first option (**Enter the name of a database**) if you specified an instance name.
7. In the Install Using section, select one of the following methods of authentication:
   - **Windows Authentication** . This option uses your current credentials (that is, the Windows credentials you used to log on to the computer on which you are installing Workload Balancing). To select this option, your current Windows credentials must have been added as a login to the SQL Server database server (instance).
   - **SQL Server Authentication** . To select this option, you must have configured SQL Server to support Mixed Mode authentication.

     **Note**

     Xen.org recommends clicking Test Connect to ensure Setup can use the credentials you provided to contact the database server.

8. In the **Database Information** page, select **Install a new Workload Balancing data store** and type the name you want to assign to the Workload Balancing database in SQL Server. The default database name is `WorkloadBalancing`.
9. In the **Web Service Host Account Information** page, select **HTTPS end point** (selected by default). Edit the port number, if necessary; the port is set to 8012 by default.

   **Note**

   If you are using Workload Balancing with Xen Cloud Platform, you must select HTTPS end points. Xen Cloud Platform can only communicate with the Workload Balancing feature over SSL/TLS. If you change the port here, you must also change it on Xen

Cloud Platform using either the **Configure Workload Balancing** wizard or the xe commands.

10. For the account (on the Workload Balancing server) that Xen Cloud Platform will use to connect to Workload Balancing, select the authorization type, **User** or **Group**, and type one of the following :

    - **User name**. Enter the name of the account you created for Xen Cloud Platform (for example, workloadbalancing_user).
    - **Group name**. Enter the group name for the account you created. Specifying a group name lets you specify a group of users that have been granted permission to connect to the Web Service Host on the Workload Balancing server. Specifying a group name lets more than one person in your organization log on to Workload Balancing with their own credentials. (Otherwise, you will need to provide all users with the same set of credentials to use for Workload Balancing.)

    Specifying the authorization type lets Workload Balancing recognize the Xen Cloud Platform's connection. For more information, see the section called “Authorization for Workload Balancing ”. You do not specify the password until you configure Workload Balancing.

11. In the **SSL/TLS Certificate** page, select one of the following certificate options:

    - **Select existing certificate from a Trusted Authority**. Specifies a certificate you generated from a Trusted Authority before Setup. Click **Browse** to navigate to the certificate.
    - **Create a self-signed certificate with subject name**. Setup creates a self-signed certificate for the Workload Balancing server. Delete the certificate-chain text and enter a subject name.
    - **Export this certificate for import into the certificate store on Xen Cloud Platform**. If you want to import the certificate into the Trusted Root Certification Authorities store on the computer running Xen Cloud Platform, select this check box. Enter the full path and file name where you want the certificate saved.

12. Click **Install.**

## To install the data store separately

The following procedure installs the Workload Balancing data store only:

1. From any server with network access to the database, launch the **Workload Balancing Setup** wizard from Autorun.exe, and select the **WorkloadBalancing** installation option.
2. After the initial **Welcome** page appears, click **Next.**
3. In the **Setup Type** page, select **Workload Balancing Database Only**, and click **Next**.

   This option lets you install the Workload Balancing data store only.

   After you click **Next**, Workload Balancing Setup verifies that your system has the correct prerequisites.
4. Accept the **End-User License Agreement**, and click **Next.**
5. In the **Component Selection** page, accept the default installation and click **Next.**

This option creates and configures a database for the Workload Balancing data store.

6. In the **Database Server** page, in the **SQL Server Selection** section, select one of the following:
   - **Enter the name of a database server**. Lets you type the name of the database server that will host the data store. Use this option to specify an instance name.

     **Note**

     If you installed SQL Express and specified an instance name, append the server name with **`\yourinstancename`**. If you installed SQL Express without specifying an instance name, append the server name with **`\sqlexpress`**.

   - **Choose an existing database server**. Lets you select the database server from a list of servers Workload Balancing Setup detected on your network.
7. In the **Install Using** section, select one of the following methods of authentication:
   - **Windows Authentication**. This option uses your current credentials (that is, the Windows credentials you used to log on to the computer on which you are installing Workload Balancing). To select this option, your current Windows credentials must have been added as a login to the SQL Server database server (instance).
   - **SQL Server Authentication**. To select this option, you must have configured SQL Server to support Mixed Mode authentication.

     **Note**

     Xen.org recommends clicking **Test Connect** to ensure Setup can use the credentials you provided to contact the database server.

8. In the **Database Information** page, select **Install a new Workload Balancing data store** and type the name you want to assign to the Workload Balancing database in SQL Server. The default database name is `WorkloadBalancing`.
9. Click **Install** to install the data store.

## To install Workload Balancing components separately

The following procedure installs Workload Balancing services on separate computers:

1. Launch the **Workload Balancing Setup** wizard from Autorun.exe, and select the **WorkloadBalancing** installation option.
2. After the initial **Welcome** page appears, click **Next.**
3. In the **Setup Type** page, select **Workload Balancing Server Services and Database**.

   This option lets you install Workload Balancing, including the Web Services Host, Analysis Engine, and Data Collection Manager services.

   Workload Balancing Setup verifies that your system has the correct prerequisites.
4. Accept the **End-User License Agreement**, and click **Next.**
5. In the **Component Selection** page, select the services you want to install:

- **Services** .
- **Data Collection Manager**. Installs the Data Collection Manager service, which collects data from the virtual machines and their hosts and writes this data to the data store.
- **Analysis Engine**. Installs the Analysis Engine service, which monitors resource pools and recommends optimizations by evaluating the performance metrics the data collector gathered.
- **Web Service Host**. Installs the service for the Web Service Host, which facilitates communications between Xen Cloud Platform and the Analysis Engine.

  If you enable the Web Service Host component, Setup prompts you for a security certificate. You can either use the self-signed certificate Workload Balancing Setup provides or specify a certificate from a Trusted Authority.

6. In the **Database Server** page, in the **SQL Server Selection** section, select one of the following:
   - **Enter the name of a database server**. Lets you type the name of the database server that is hosting the data store.

     **Note**

     If you installed SQL Express and specified an instance name, append the server name with **`\yourinstancename`**. If you installed SQL Express without specifying an instance name, append the server name with **`\sqlexpress`**.

   - **Choose an existing database server**. Lets you select the database server from a list of servers Workload Balancing Setup detected on your network.

   **Note**

   Xen.org recommends clicking **Test Connect** to ensure Setup can use the credentials you provided to contact the database server successfully.

7. In the **Web Service Information** page, select **HTTPS end point** (selected by default) and edit the port number, if necessary. The port is set to 8012 by default.

   **Note**

   If you are using Workload Balancing with Xen Cloud Platform, you must select HTTPS end points. Xen Cloud Platform can only communicate with the Workload Balancing feature over SSL/TLS. If you change the port here, you must also change it on Xen Cloud Platform using either the **Configure Workload Balancing** wizard or the xe commands.

8. For the account (on the Workload Balancing server) that Xen Cloud Platform will use to connect to Workload Balancing, select the authorization type, **User** or **Group**, and type one of the following:
   - **User name**. Enter the name of the account you created for Xen Cloud Platform (for example, `workloadbalancing_user`).

- **Group name**. Enter the group name for the account you created. Specifying a group name lets more than one person in your organization log on to Workload Balancing with their own credentials. (Otherwise, you will need to provide all users with the same set of credentials to use for Workload Balancing.)
- Specifying the authorization type lets Workload Balancing recognize the Xen Cloud Platform's connection. For more information, see the section called “Authorization for Workload Balancing ”. You do not specify the password until you configure Workload Balancing.

9. In the **SSL/TLS Certificate** page, select one of the following certificate options:
   - **Select existing certificate from a Trusted Authority**. Specifies a certificate you generated from a Trusted Authority before Setup. Click **Browse** to navigate to the certificate.
   - **Create a self-signed certificate with subject name**. Setup creates a self-signed certificate for the Workload Balancing server. To change the name of the certificate Setup creates, type a different name.
   - **Export this certificate for import into the certificate store on Xen Cloud Platform**. If you want to import the certificate into the Trusted Root Certification Authorities store on the computer running Xen Cloud Platform, select this check box. Enter the full path and file name where you want the certificate saved.
10. Click **Install**.

## To verify your Workload Balancing installation

Workload Balancing Setup does not install an icon in the Windows Start menu. Use this procedure to verify that Workload Balancing installed correctly before trying to connect to Workload Balancing with the Workload Balancing Configuration wizard.

1. Verify Windows **Add or Remove Programs** (Windows XP) lists **Xen.org Workload Balancing** in its in the list of currently installed programs.
2. Check for the following services in the Windows **Services** panel:
   - Xen.org WLB Analysis Engine
   - Xen.org WLB Data Collection Manager
   - Xen.org WLB Web Service Host

   All of these services must be started and running before you start configuring Workload Balancing.
3. If Workload Balancing appears to be missing, check the installation log to see if it installed successfully:
   - If you used the Setup wizard, the log is at %Documents and Settings%\username\Local Settings\Temp\msibootstrapper2CSM_MSI_Install.log (by default). On Windows Vista and Windows Server 2008, this log is at %Users%\username\AppData\Local\Temp\msibootstrapper2CSM_MSI_Install.log. User name is the name of the user logged on during installation.
   - If you used the Setup properties (Msiexec), the log is at <C>:\log.txt (by default) or wherever you specified for Setup to create it.

# Windows Installer Commands for Workload Balancing

The Workload Balancing installation supports using the Msiexec command for Setup. The Msiexec command lets you install, modify, and perform operations on Windows Installer (.msi) packages from the command line.

Set properties by adding `Property="value"` on the command line after other switches and parameters.

The following sample command line performs a full installation of the Workload Balancing Windows Installer package and creates a log file to capture information about this operation.

```
msiexec.exe /I C:\path-to-msi\workloadbalancingx64.msi /quiet
PREREQUISITES_PASSED="1"
DBNAME="WorkloadBalancing1"
DATABASESERVER="WLB-DB-SERVER\INSTANCENAME"
HTTPS_PORT="8012"
WEBSERVICE_USER_CB="0"
USERORGROUPACCOUNT="domain\WLBgroup"
CERT_CHOICE="0"
CERTNAMEPICKED="cn=wlb-cert1"
EXPORTCERT=1
EXPORTCERT_FQFN="C:\Certificates\WLBCert.cer"
INSTALLDIR="C:\Program Files\Xen.org\WLB"
ADDLOCAL="Database,Complete,Services,DataCollection,
Analysis_Engine,DWM_Web_Service" /l*v log.txt
```

There are two Workload Balancing Windows Installer packages: workloadbalancing.msi and workloadbalancingx64.msi. If you are installing Workload Balancing on a 64-bit operating system, specify workloadbalancingx64.msi.

To see if Workload Balancing Setup succeeded, see the section called “To verify your Workload Balancing installation”.

**Important**

Workload Balancing Setup does not provide error messages if you are installing Workload Balancing using Windows Installer commands if the system is missing prerequisites. Instead, installation fails.

## ADDLOCAL

### Definition

Specifies one or more Workload Balancing features to install. The values of ADDLOCAL are Workload Balancing components and services.

### Possible values

- **Database**. Installs the Workload Balancing data store.
- **Complete**. Installs all Workload Balancing features and components.

- **Services**. Installs all Workload Balancing services, including the Data Collection Manager, the Analysis Engine, and the Web Service Host service.
- **DataCollection**. Installs the Data Collection Manager service.
- **Analysis_Engine**. Installs the Analysis Engine service.
- **DWM_Web_Service**. Installs the Web Service Host service.

### Default value

Blank

### Remarks

- Separate entries by commas.
- The values must be installed locally.
- You must install the data store on a shared or dedicated server before installing other services.
- You can only install services standalone, without installing the database simultaneously, if you have a Workload Balancing data store installed and specify it in the installation script using and for the database type. See the section called “DBNAME” and the section called “DATABASESERVER” for more information.

## CERT_CHOICE

### Definition

Specifies for Setup to either create a certificate or use an existing certificate.

### Possible values

- **0**. Specifies for Setup to create a new certificate.
- **1**. Specifies an existing certificate.

### Default value

1

### Remarks

- You must also specify `CERTNAMEPICKED`. See the section called “CERTNAMEPICKED” for more information.

## CERTNAMEPICKED

### Definition

Specifies the subject name when you use Setup to create a self-signed SSL/TLS certificate. Alternatively, this specifies an existing certificate.

### Possible values

**cn**. Use to specify the subject name of certificate to use or create.

### Example

`cn=<wlb-kirkwood>`, where *<wlb-kirkwood>* is the name you are specifying as the name of the certificate to create or the certificate you want to select.

### Default value

Blank.

### Remarks

You must specify this parameter with the `CERT_CHOICE` parameter. See the section called "CERT_CHOICE" for more information.

## DATABASESERVER

### Definition

Specifies the database, and its instance name, where you want to install the data store. You can also use this property to specify an existing database that you want to use or upgrade.

### Possible values

User defined.

---

**Note**

If you specified an instance name when you installed SQL Server or SQL Express, append the server name with **`\yourinstancename`**. If you installed SQL Express without specifying an instance name, append the server name with **`\sqlexpress`**.

---

### Default value

Local

### Example

**`DATABASESERVER="WLB-DB-SERVER\SQLEXPRESS"`**, where `WLB-DB-SERVER` is the name of your database server and `SQLEXPRESS` is the name of the database instance.

### Remarks

- Required property for all installations.
- Whether installing a database or connecting to an existing data store, you must specify this property with DBNAME.

- Even if you are specifying a database on the same computer as you are performing Setup, you still must define the name of the database.
- When you specify DATABASESERVER, in some circumstances, you must specify also the section called “WINDOWS_AUTH” and its accompanying properties.

## DBNAME

### Definition

The name of the Workload Balancing database that Setup will create or upgrade during installation.

### Possible values

User defined.

### Default value

WorkloadBalancing

### Remarks

- Required property for all installations. You must set a value for this property.
- Whether connecting to or installing a data store, you must specify this property with DATABASESERVER.
- Even if you are specifying a database on the same computer as you are performing Setup, you still must define the name of the database.
- Localhost is not a valid value.

## DBUSERNAME

### Definition

Specifies the user name for the Windows or SQL Server account you are using for database authentication during Setup.

### Possible values

User defined.

### Default value

Blank

### Remarks

- This property is used with `WINDOWS_AUTH` (see the section called “WINDOWS_AUTH”) and `DBPASSWORD` (see the section called “DBPASSWORD”.)

- Because you specify the server name and instance using the section called “DATABASESERVER”, do not qualify the user name.

## DBPASSWORD

### Definition

Specifies the password for the Windows or SQL Server account you are using for database authentication during Setup.

### Possible values

User defined.

### Default value

Blank.

### Remarks

- Use this property with the parameters documented in the section called “WINDOWS_AUTH” and the section called “DBUSERNAME”.

## EXPORTCERT

### Definition

Set this value to export an SSL/TLS certificate from the server on which you are installing Workload Balancing. Exporting the certificate lets you import it into the certificate stores of computers running Xen Cloud Platform.

### Possible values

- **0**. Does not exports the certificate.
- **1**. Exports the certificate and saves it to the location of your choice with the file name you specify using EXPORTCERT_FQFN.

### Default value

0

### Remarks

- Use with the section called “EXPORTCERT_FQFN”, which specifies the file name and path.
- Setup does not require this property to run successfully. (That is, you do not have to export the certificate.)

- This property lets you export self-signed certificates that you create during Setup as well as certificates that you created using a Trusted Authority.

## EXPORTCERT_FQFN

### Definition

Set to specify the path (location) and the file name you want Setup to use when exporting the certificate.

### Possible values

The fully qualified path and file name to which to export the certificate. For example, C:\Certificates\WLBCert.cer.

### Default value

Blank.

### Remarks

Use this property with the parameter documented in the section called “EXPORTCERT”.

## HTTPS_PORT

### Definition

Use this property to change the default port over which Workload Balancing (the Web Service Host service) communicates with Xen Cloud Platform.

Specify this property when you are running Setup on the computer that will host the Web Service Host service. This may be either the Workload Balancing computer, in a one-server deployment, or the computer hosting the services.

### Possible values

User defined.

### Default value

8012

### Remarks

- If you set a value other than the default for this property, you must also change the value of this port in Xen Cloud Platform, which you can do with the **Configure Workload Balancing** wizard. The port number value specified during Setup and in the **Configure Workload Balancing** wizard must match.

## INSTALLDIR

### Definition

`Installation directory`, where `Installation directory` is the location where the Workload Balancing software is installed.

### Possible values

User configurable

### Default value

C:\Program Files\Xen.org

## PREREQUISITES_PASSED

### Definition

You must set this property for Setup to continue. When enabled (PREREQUISITES_PASSED = 1), Setup skips checking preinstallation requirements, such as memory or operating system configurations, and lets you perform a command-line installation of the server.

### Possible values

- **1**. Indicates for Setup to not check for preinstallation requirements on the computer on which you are running Setup. You must set this property to **1** or Setup fails.

### Default value

0

### Remarks

- This is a required value.

## RECOVERYMODEL

### Definition

Specifies the SQL Server database recovery model.

### Possible values

- **SIMPLE**. Specifies the SQL Server Simple Recovery model. Lets you recover the database from the end of any backup. Requires the least administration and consumes the lowest amount of disk space.

- **FULL**. Specifies the Full Recovery model. Lets you recover the database from any point in time. However, this model uses consumes the largest amount of disk space for its logs.
- **BULK_LOGGED**. Specifies the Bulk-Logged Recovery model. Lets you recover the database from the end of any backup. This model consumes less logging space than the Full Recovery model. However, this model provides more protection for data than the Simple Recovery model.

### Default value

SIMPLE

### Remarks

For more information about SQL Server recovery models, see the Microsoft's MSDN Web site and search for "Selecting a Recovery Model."

## USERORGROUPACCOUNT

### Definition

Specifies the account or group name that corresponds with the account Xen Cloud Platform will use when it connects to Workload Balancing. Specifying the name lets Workload Balancing recognize the connection.

### Possible values

- **User name**. Specify the name of the account you created for Xen Cloud Platform (for example, **`workloadbalancing_user`**).
- **Group name**. Specify the group name for the account you created. Specifying a group name lets more than one person in your organization log on to Workload Balancing with their own credentials. (Otherwise, you will have to provide all users with the same set of credentials to use for Workload Balancing.)

### Default value

Blank.

### Remarks

- This is a required parameter. You must use this parameter with the section called “WEBSERVICE_USER_CB ”.
- To specify this parameter, you must create an account on the Workload Balancing server before running Setup. For more information, see the section called “Authorization for Workload Balancing ”.
- This property does not require specifying another property for the password. You do not specify the password until you configure Workload Balancing.

## WEBSERVICE_USER_CB

### Definition

Specifies the authorization type, user account or group name, for the account you created for Xen Cloud Platform before Setup. For more information, see the section called “Authorization for Workload Balancing ”.

### Possible values

- **0**. Specifies the type of data you are specifying with USERORGROUPACCOUNT corresponds with a group.
- **1**. Specifies the type of data you are specifying with USERORGROUPACCOUNT corresponds with a user account.

### Default value

0

### Remarks

- This is a required property. You must use this parameter with the section called “USERORGROUPACCOUNT”.

## WINDOWS_AUTH

### Definition

Lets you select the authentication mode, either Windows or SQL Server, when connecting to the database server during Setup. For more information about database authentication during Setup, see SQL Server Database Authentication Requirements.

### Possible values

- **0**. SQL Server authentication
- **1**. Windows authentication

### Default value

1

### Remarks

- If you are logged into the server on which you are installing Workload Balancing with Windows credentials that have an account on the database server, you do not need to set this property.

- If you specify WINDOWS_AUTH, you must also specify DBPASSWORD if you want to specify an account other than the one you are logged onto the server on which you are running Setup.
- The account you specify must be a login on the SQL Server database with **sysadmin** privileges.

# Initializing and Configuring Workload Balancing

Following Workload Balancing Setup, you must configure and enable (that is, initialize) Workload Balancing on each resource pool you want to monitor before Workload Balancing can gather data for that pool.

Before initializing Workload Balancing, configure your antivirus software to exclude Workload Balancing folders, as described in the section called “Configuring Antivirus Software”.

After the initial configuration, the **Initialize** button on the **WLB** tab changes to a **Disable** button. This is because after initialization you cannot modify the Workload Balancing server a resource pool uses without disabling Workload Balancing on that pool and then reconfiguring it. For information, see ???.

---

**Important**

Following initial configuration, Xen.org strongly recommends you evaluate your performance thresholds as described in ???. It is critical to set Workload Balancing to the correct thresholds for your environment or its recommendations might not be appropriate.

---

You can use the xe commands to initialize Workload Balancing or modify the configuration settings.

## Initialization Overview

Initial configuration requires that you:

1. **Specify the Workload Balancing server** you want the resource pool to use and its port number.
2. **Specify the credentials** for communications, including the credentials:
   - Xen Cloud Platform will use to connect to the Workload Balancing server
   - Workload Balancing will use to connect to Xen Cloud Platform

   For more information, see the section called “Authorization for Workload Balancing ”
3. **Change the optimization mode**, if desired, from Maximum Performance, the default setting, to Maximize Density. For information about the placement strategies, see ???.
4. **Modify performance thresholds**, if desired. You can modify the default utilization values and the critical thresholds for resources. For information about the performance thresholds, see ???.
5. **Modify metric weighting**, if desired. You can modify the importance Workload Balancing assigns to metrics when it evaluates resource usage. For information about the performance thresholds, see ???.

## To initialize Workload Balancing

Use this procedure to enable and perform the initial configuration of Workload Balancing for a resource pool.

Before the Workload Balancing feature can begin collecting performance data, the Xen Cloud Platforms you want to balance must be part of a resource pool. To complete this wizard, you need the:

- IP address (or NetBIOS name) and (optionally) port of the Workload Balancing server
- Credentials for the resource pool you want Workload Balancing to monitor
- Credentials for the account you created on the Workload Balancing server

## To edit the Workload Balancing configuration for a pool

After initialization, you can use this procedure to edit the Workload Balancing performance thresholds and placement strategies for a specific resource pool.

## Authorization for Workload Balancing

When you are configuring a Xen Cloud Platform resource pool to use Workload Balancing, you must specify credentials for two accounts:

- **User Account for Workload Balancing to connect to Xen Cloud Platform**. Workload Balancing uses a Xen Cloud Platform user account to connect to Xen Cloud Platform. You provide Workload Balancing with this account's credentials when you run the **Configure Workload Balancing** wizard. Typically, you specify the credentials for the pool (that is, the pool master's credentials).
- **User Account for Xen Cloud Platform to Connect to Workload Balancing**. Xen Cloud Platform communicates with the Web Service Host using the user account you created before Setup.

  During Workload Balancing Setup, you specified the authorization type (a single user or group) and the user or group with permissions to make requests from the Web Service Host service.

  During configuration, you must provide Xen Cloud Platform with this account's credentials when you run the **Configure Workload Balancing** wizard.

# Configuring Antivirus Software

By default, most antivirus programs are configured to scan all files on the hard disk. If an antivirus program scans the frequently active Workload Balancing database, it impedes or slows down the normal operation of Workload Balancing. Consequently, you must configure antivirus software running on your Workload Balancing servers to exclude specific processes and files. Xen.org recommends configuring your antivirus software to exclude these folders before you initialize Workload Balancing and begin collecting data.

To configure antivirus software on the servers running Workload Balancing components:

- Exclude the following folder, which contains the Workload Balancing log:

  On Windows XP and Windows Server 2003: %Documents and Settings%\All Users\Application Data\Xen.org\Workload Balancing\Data\Logfile.log

  On Windows Vista and Windows Server 2008: %Program Data%\Xen.org\Workload Balancing\Data\Logfile.log.
- Exclude the SQL Server database folder. For example:

  On SQL Server: %Program Files%\Microsoft SQL Server\MSSQL\Data\

  On SQL Server Express: %Program Files%\Microsoft SQL Server\MSSQL10.SQLEXPRESS\MSSQL\Data\

  These paths may vary according to your operating system and SQL Server version.

---

**Note**

These paths and file names are for 32-bit default installations. Use the values that apply to your installation. For example, paths for 64-bit edition files might be in the %Program Files (x86)% folder.

---

# Chapter 6. Backup and recovery

This chapter presents the functionality designed to give you the best chance to recover your Xen Cloud Platform from a catastrophic failure of hardware or software, from lightweight metadata backups to full VM backups and portable SRs.

## Backups

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

### To backup pool metadata

1. Run the command:

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

2. Run the command:

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

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

### To backup host configuration and software

- Run the command:

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

---

**Note**

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

---

### To backup a VM

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

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

**Note**

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

**Warning**

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

### To backup VM metadata only

- Run the command:

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

## Full metadata backup and disaster recovery (DR)

This section introduces the concept of Portable Storage Repositories (Portable SRs), and explains how they work and how to use them as part of a DR strategy.

## DR and metadata backup overview

Xen Cloud Platform 0.1 introduces the concept of Portable SRs. Portable SRs contain all of the information necessary to recreate all the Virtual Machines (VMs) with Virtual Disk Images (VDIs) stored on the SR after re-attaching the SR to a different host or pool. Portable SRs can be used when regular maintenance or disaster recovery requires manually moving a SR between pools or standalone hosts.

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

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

**Note**

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

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

The metadata backup and restore feature works at the command-line level and the same functionality is also supported in xsconsole.

## Backup and restore using xsconsole

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

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

On the Xen Cloud Platform host menu-driven text console, under the **Backup, Update and Restore** menu there are options which provide a more user-friendly interface to these scripts. The operations should only be performed on the pool master. You can use these menu options to perform 3 operations:

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

For automating this scripting, there are some commands in the control domain which provide an interface to metadata backup and restore at a lower level than the menu options:

- **xe-backup-metadata** provides an interface to create the backup VDIs (with the -c flag), and also to attach the metadata backup and examine its contents.

- **xe-restore-metadata** can be used to probe for a backup VDI on a newly attached SR, and also selectively reimport VM metadata to recreate the associations between VMs and their disks.

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

## Moving SRs between hosts and Pools

The metadata backup and restore options can be run as scripts in the control domain or through the **Backup, Restore, and Update** menu option in the xsconsole. All other actions, such as detaching the SR from the source host and attaching it to the destination host, can be performed using the xe CLI.

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

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

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

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

## VM Snapshots

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

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

The snapshotting operation is a 2 step process:

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

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

## Regular Snapshots

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

## Quiesced Snapshots

Quiesced snapshots take advantage of the Windows Volume Shadow Copy Service (VSS) to generate application consistent point-in-time snapshots. The VSS framework helps VSS-aware applications (for example Microsoft Exchange or Microsoft SQL Server) flush data to disk and prepare for the snapshot before it is taken.

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

Xen Cloud Platform supports quiesced snapshots on Windows Server 2003 and Windows Server 2008 for both 32-bit and 64-bit variants. Windows 2000, Windows XP and Windows Vista are not supported. Snapshot is supported on all storage types, though for the LVM-based storage types the storage repository must have been upgraded if it was created on a previous version of Xen Cloud Platform and the volume must be in the default format (`type=raw` volumes cannot be snapshotted).

---

**Note**

Do not forget to install the Xen VSS provider in the Windows guest in order to support VSS. This is done using the `install-XenProvider.cmd` script provided with the Windows PV drivers. More details can be found in the *Virtual Machine Installation Guide* in the Windows section.

---

In general, a VM can only access VDI snapshots (not VDI clones) of itself using the VSS interface. There is a flag that can be set by the Xen Cloud Platform administrator whereby adding an attribute of *`snapmanager=true`* to the VM's *`other-config`* allows that VM to import snapshots of VDIs from other VMs.

**Warning**

This opens a security vulnerability and should be used with care. This feature allows an administrator to attach VSS snapshots using an in-guest transportable snapshot ID as generated by the VSS layer to another VM for the purposes of backup.

*VSS quiesce timeout*: the Microsoft VSS quiesce period is set to a non-configurable value of 10 seconds, and it is quite probable that a snapshot may not be able to complete in time. If, for example the XAPI daemon has queued additional blocking tasks such as an SR scan, the VSS snapshot may timeout and fail. The operation should be retried if this happens.

**Note**

The more VBDs attached to a VM, the more likely it is that this timeout may be reached. Xen.org recommends attaching no more that 2 VBDs to a VM to avoid reaching the timeout. However, there is a workaround to this problem. The probability of taking a successful VSS based snapshot of a VM with more than 2 VBDs can be increased manifold, if all the VDIs for the VM are hosted on different SRs.

*VSS snapshot all the disks attached to a VM*: in order to store all data available at the time of a VSS snapshot, the XAPI manager will snapshot all disks and the VM metadata associated with a VM that can be snapshotted using the Xen Cloud Platform storage manager API. If the VSS layer requests a snapshot of only a subset of the disks, a full VM snapshot will not be taken.

*vm-snapshot-with-quiesce produces bootable snapshot VM images*: To achieve this, the Xen Cloud Platform VSS hardware provider makes snapshot volumes writable, including the snapshot of the boot volume.

*VSS snap of volumes hosted on dynamic disks in the Windows Guest*: The vm-snapshot-with-quiesce CLI and the Xen Cloud Platform VSS hardware provider do not support snapshots of volumes hosted on dynamic disks on the Windows VM.

## Taking a VM snapshot

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

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

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

## VM Rollback

### Restoring a VM to snapshot state

**Note**

Restoring a VM will not preserve the original VM UUID or MAC address.

1. Note the name of the snapshot
2. Note the MAC address of the VM
3. Destroy the VM:
   a. Run the **vm-list** command to find the UUID of the VM to be destroyed:

   ```
   xe vm-list
   ```

   b. Shutdown the VM:

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

   c. Destroy the VM:

   ```
   xe vm-destroy uuid=<vm_uuid>
   ```
4. Create a new VM from the snapshot:

   ```
   xe vm-install new-name-label=<vm_name_label> template=<template_name>
   ```
5. Start the VM:

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

# Coping with machine failures

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

## Member failures

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

- Repair the dead host (e.g. by physically rebooting it). When the connection to the member is restored, the master will mark the member as alive again.
- Shutdown the host and instruct the master to forget about the member node using the **xe host-forget** CLI command. Once the member has been forgotten, all the VMs which were running there will be marked as offline and can be restarted on other Xen Cloud

Platform hosts. Note it is *very* important to ensure that the Xen Cloud Platform host is actually offline, otherwise VM data corruption might occur. Be careful not to split your pool into multiple pools of a single host by using **xe host-forget**, since this could result in them all mapping the same shared storage and corrupting VM data.

**Warning**

- If you are going to use the forgotten host as a Xen Cloud Platform host again, perform a fresh installation of the Xen Cloud Platform software.
- Do not use **xe host-forget** command if HA is enabled on the pool. Disable HA first, then forget the host, and then re-enable HA.

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

**Warning**

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

## Master failures

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

1. The members realize that communication has been lost and each tries to reconnect for sixty seconds.
2. Each member then puts itself into emergency mode, whereby the member Xen Cloud Platform 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 re-establishes communication with its members, the members leave emergency mode, and operation returns to normal.

If the master is really dead, choose one of the members and run the command **xe pool-emergency-transition-to-master** on it. Once it has become the master, run 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 Xen Cloud Platform host software, and add it to the pool. Since the Xen Cloud Platform hosts in the pool are enforced to be homogeneous, there is no real need to make the replaced server the master.

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

## Pool failures

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

### To restore a completely failed pool

1. Install a fresh set of hosts. Do not pool them up at this stage.
2. For the host nominated as the master, restore the pool database from your backup using the **xe pool-restore-database** (see the section called “pool-restore-database”) command.
3. Perform a pool join operation on the remaining freshly installed member hosts, and start up your VMs on the appropriate hosts.

## Coping with Failure due to Configuration Errors

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

### To restore host software and configuration

1. Run the command:

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

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

## Physical Machine failure

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

---

### Warning

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

```
xe vm-reset-powerstate vm=<vm_uuid> --force
```

VMs can then be restarted using the CLI.

---

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

1. Run the commands:

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

2. If the commands succeed, restart the VMs.

### To restore a pool with all hosts failed

1. Run the command:

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

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

2. If the target machine has a different view of the storage (for example, a block-mirror with a different IP address) than the original machine, modify the storage configuration using the **pbd-destroy** command and then the **pbd-create** command to recreate storage configurations. See the section called “PBD commands” for documentation of these commands.
3. If you have created a new storage configuration, use **pbd-plug** to use the new configuration.
4. Restart all VMs.

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

1. Run the command:

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

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

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

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

# Chapter 7. Monitoring and managing Xen Cloud Platform

Xen Cloud Platform provide access to alerts that are generated when noteworthy things happen.

## Alerts

Xen Cloud Platform generates alerts for the following events.

Configurable Alerts:

- New Xen Cloud Platform patches available
- New Xen Cloud Platform version available

Alerts generated by Xen Cloud Platform:

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

## Customizing Alerts

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

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

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

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

### Valid VM Elements

name
: what to call the variable (no default). If the name value is one of `cpu_usage`, `network_usage`, or `disk_usage` the `rrd_regex` and `alarm_trigger_sense` parameters are not required as defaults for these values will be used.

alarm_priority
: the priority of the messages generated (default `5`)

alarm_trigger_level
: level of value that triggers an alarm (no default)

alarm_trigger_sense
: `high` if *`alarm_trigger_level`* is a maximum value otherwise `low` if the *`alarm_trigger_level`* is a minimum value. (default `high`)

alarm_trigger_period
: number of seconds that values above or below the alarm threshold can be received before an alarm is sent (default `60`)

alarm_auto_inhibit_period
: number of seconds this alarm disabled after an alarm is sent (default `3600`)

consolidation_fn
: how to combine variables from rrd_updates into one value (default is `sum` - other choice is `average`)

rrd_regex
: regular expression to match the names of variables returned by the **xe vm-data-source-list uuid=<***vmuuid***>** command that should be used to compute the statistical value. This parameter has defaults for the named variables `cpu_usage`, `network_usage`, and `disk_usage`. If specified, the values of all items returned by **xe vm-data-source-list** whose names match the specified regular expression will be consolidated using the method specified as the *`consolidation_fn`*.

### Valid Host Elements

name
: what to call the variable (no default)

alarm_priority
: the priority of the messages generated (default `5`)

alarm_trigger_level
: level of value that triggers an alarm (no default)

alarm_trigger_sense
: `high` if *`alarm_trigger_level`* is a maximum value otherwise `low` if the *`alarm_trigger_level`* is a minimum value. (default `high`)

alarm_trigger_period
: number of seconds that values above or below the alarm threshold can be received before an alarm is sent (default `60`)

alarm_auto_inhibit_period
: number of seconds this alarm disabled after an alarm is sent (default `3600`)

consolidation_fn
: how to combine variables from **rrd_updates** into one value (default `sum` - other choice is `average`)

rrd_regex
: regular expression to match the names of variables returned by the **xe vm-data-source-list uuid=<*vmuuid*>** command that should be used to compute the statistical value. This parameter has defaults for the named variables `cpu_usage` and `network_usage`. If specified, the values of all items returned by **xe vm-data-source-list** whose names match the specified regular expression will be consolidated using the method specified as the *`consolidation_fn`*.

## Configuring Email Alerts

Alerts generated from Xen Cloud Platform can also be automatically e-mailed to the resource pool administrator. To configure this, specify the email address and SMTP server:

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

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

```
pool:other-config:mail-min-priority=<level>
```

The default priority level is `5`.

**Note**

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

# Determining throughput of physical bus adapters

For FC, SAS and iSCSI HBAs you can determine the network throughput of your PBDs using the following procedure.

### To determine PBD throughput

1. List the PBDs on a host.
2. Determine which LUNs are routed over which PBDs.
3. For each PBD and SR, list the VBDs that reference VDIs on the SR.
4. For all active VBDs that are attached to VMs on the host, calculate the combined throughput.

For iSCSI and NFS storage, check your network statistics to determine if there is a throughput bottleneck at the array, or whether the PBD is saturated.

# Chapter 8. Command line interface

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

The xe command line interface is installed by default on Xen Cloud Platform hosts. A stand-alone remote CLI is also available for Linux.

On Windows, the `xe.exe` CLI executable is installed.

To use it, open a Windows Command Prompt and change directories to the directory where the file resides 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-0.1-@BUILD_NUMBER@.i386.rpm` on the Linux Pack CD, as follows:

```
rpm -ivh xe-cli-0.1-@BUILD_NUMBER@.i386.rpm
```

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

```
xe help command
```

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

```
xe help
```

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

```
xe help --all
```

## Basic xe syntax

The basic syntax of all Xen Cloud Platform 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 Xen Cloud Platform host (defaults to 443).

*Example:* On the local Xen Cloud Platform host:

```
xe vm-list
```

*Example:* On the remote Xen Cloud Platform host:

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

Shorthand syntax is also available for remote connection arguments:

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

*Example:* On a remote Xen Cloud Platform 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 Xen Cloud Platform host that are run on a remote Xen Cloud Platform host, you could do the following:

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

This command means that you will not need to specify the remote Xen Cloud Platform host parameters anymore, 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 Xen Cloud Platform host, which is disabled by default.

## Special characters and syntax

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

argument=value

Unless the value includes spaces, do not use quotes. 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 Xen Cloud Platform 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, pressing **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 Xen Cloud Platform host, tab completion does not normally work. However if you put the server, username, and password in an environment variable called XE_EXTRA_ARGS on the machine from which you are entering the commands, tab completion is enabled. See the section called “Basic xe syntax” for details.

## Command types

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

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

where <*class*> is one of:

- bond
- console
- host
- host-crashdump
- host-cpu
- network
- patch
- pbd
- pif
- pool
- sm
- sr

- task
- template
- vbd
- vdi
- vif
- vlan
- vm

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

## Parameter types

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

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

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

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

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

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

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

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

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

**Note**

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

## Low-level param commands

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

<class>-param-list uuid=<uuid>
: Lists all of the parameters and their associated values. Unlike the *class*-list command, this will list the values of "expensive" fields.

<class>-param-get uuid=<uuid> param-name=<parameter> [param-key=<key>]
: Returns the value of a particular parameter. If the parameter is a map, specifying the param-key will get the value associated with that key in the map. If param-key is not specified, or if the parameter is a set, it will return a string representation of the set or map.

<class>-param-set uuid=<uuid> param=<value>...
: Sets the value of one or more parameters.

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

<class>-param-remove uuid=<uuid> param-name=<parameter> param-key=<key>
: Removes either a key/value pair from a map, or a key from a set.

<class>-param-clear uuid=<uuid> param-name=<parameter>
: Completely clears a set or a map.

## Low-level list commands

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

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

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

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

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

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

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

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 objects that match all of the specified constraints. For example:

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

This command 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 Xen Cloud Platform host with three VMs installed gives the three UUIDs of the VMs, for example:

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

# xe command reference

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

## Bonding commands

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

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

### Bond parameters

Bonds have the following parameters:

| Parameter Name | Description | Type |
|---|---|---|
| uuid | unique identifier/object reference for the bond | read only |
| master | UUID for the master bond PIF | read only |

| Parameter Name | Description | Type |
|---|---|---|
| members | set of UUIDs for the underlying bonded PIFs | read only set parameter |

### 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 Xen Cloud Platform host, or if fewer than 2 PIFs are supplied.

### bond-destroy

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

Delete a bonded interface specified by its UUID from the Xen Cloud Platform host.

## CD commands

Commands for working with physical CD/DVD drives on Xen Cloud Platform 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 |

| Parameter Name | Description | Type |
|---|---|---|
| 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 CD drive is sharable. Default is false. | 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 |
| 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 |
| is-a-snapshot | True if this template is a CD snapshot | read only |
| snapshot_of | The UUID of the CD that this template is a snapshot of | read only |

| Parameter Name | Description | Type |
|---|---|---|
| snapshots | The UUID(s) of any snapshots that have been taken of this CD | read only |
| snapshot_time | The timestamp of the snapshot operation | read only |

### cd-list

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

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

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

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

## Console commands

Commands for working with consoles.

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

#### Console parameters

Consoles have the following parameters:

| 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 |

| Parameter Name | Description | Type |
|---|---|---|
| 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 |

# 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 (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 |

## event-wait

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

Blocks 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 other commands 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 other commands until a VM with UUID *$VM* reboots (i.e. has a different *start-time* value).

The class name can be any of the Event classes listed at the beginning of this section, and the parameters can be any of those listed in the CLI command *class*-param-list.

## Host (Xen Cloud Platform host) commands

Commands for interacting with Xen Cloud Platform host.

Xen Cloud Platform hosts are the physical servers running Xen Cloud Platform 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 Xen Cloud Platform host objects can be listed with the standard object listing command (**xe host-list**, **xe host-cpu-list**, and **xe host-crashdump-list**), and the parameters manipulated with the standard parameter commands. See the section called “Low-level param commands” for details.

### Host selectors

Several of the commands listed here have a common mechanism for selecting one or more Xen Cloud Platform hosts on which to perform the operation. The simplest is by supplying the argument *host=<uuid_or_name_label>*. Xen Cloud Platform 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 Xen Cloud Platform hosts whose *enabled* field is equal to *true*. Where multiple Xen Cloud Platform hosts are matching, and the operation can be performed on multiple Xen Cloud Platform 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 running the command **xe host-list params=all**. If no parameters to select Xen Cloud Platform hosts are given, the operation will be performed on all Xen Cloud Platform hosts.

### Host parameters

Xen Cloud Platform hosts have the following parameters:

| Parameter Name | Description | Type |
|---|---|---|
| uuid | The unique identifier/object reference for the Xen Cloud Platform host | read only |
| name-label | The name of the Xen Cloud Platform host | read/write |
| name-description | | read only |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | The description string of the Xen Cloud Platform host | |
| enabled | false if disabled which prevents any new VMs from starting on them, which prepares the Xen Cloud Platform 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 | identification of API 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 Xen Cloud Platform host can run | read only set parameter |
| other-config | A list of key/value pairs that specify additional configuration parameters for the Xen Cloud Platform host | read/write map parameter |

| Parameter Name | Description | Type |
| --- | --- | --- |
| hostname | Xen Cloud Platform host hostname | read only |
| address | Xen Cloud Platform host IP address | read only |
| supported-bootloaders | list of bootloaders that the Xen Cloud Platform host supports, for example, pygrub, eliloader | read only set parameter |
| memory-total | total amount of physical RAM on the Xen Cloud Platform 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-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 | lists 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 | lists the operations currently in process. 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 |
| patches | Set of host patches | read only set parameter |
| blobs | Binary data store | read only |

| Parameter Name | Description | Type |
|---|---|---|
| memory-free-computed | A conservative estimate of the maximum amount of memory free on a host | read only |
| ha-statefiles | The UUID(s) of all HA statefiles | read only |
| ha-network-peers | The UUIDs of all hosts that could host the VMs on this host in case of failure | read only |
| external-auth-type | Type of external authentication, for example, Active Directory. | read only |
| external-auth-service-name | The name of the external authentication service | read only |
| external-auth-configuration | Configuration information for the external authentication service. | read only map parameter |

Xen Cloud Platform hosts contain some other objects that also have parameter lists.

CPUs on Xen Cloud Platform 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 Xen Cloud Platform 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 |

| Parameter Name | Description | Type |
|---|---|---|
| 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 physical CPU family number | read only |

Crash dumps on Xen Cloud Platform hosts have the following parameters:

| Parameter Name | Description | Type |
|---|---|---|
| uuid | The unique identifier/object reference for the crashdump | read only |
| host | Xen Cloud Platform 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 |

## host-backup

`host-backup` file-name=<*backup_filename*> host=<*host_name*>

Download a backup of the control domain of the specified Xen Cloud Platform host to the machine that the command is invoked from, and save it there as a file with the name *`filename`*.

**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. The command should *only* be used from a remote off-host machine where you have space to hold the backup file.

## host-bugreport-upload

`host-bugreport-upload` [<*host-selector*>=<*host_selector_value*>...] [url=<*destination_url*>]

[http-proxy=<*http_proxy_name*>]

Generate a fresh bug report (using xen-bugtool, with all optional files included) and upload to the Xen.org Support ftp site or some other location.

The host(s) on which this operation should be performed are selected using 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 Xen.org Support ftp site.

### host-crashdump-destroy

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

Delete a host crashdump specified by its UUID from the Xen Cloud Platform host.

### host-crashdump-upload

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

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

### host-disable

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

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

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

### host-dmesg

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

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

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

### host-emergency-management-reconfigure

`host-emergency-management-reconfigure` interface=*<uuid_of_management_interface_pif>*

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

### host-enable

`host-enable` [*<host-selector>*=*<host_selector_value>*...]

Enables the specified Xen Cloud Platform hosts, which allows new VMs to be started on them.

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

### host-evacuate

`host-evacuate` [*<host-selector>*=*<host_selector_value>*...]

Live migrates all running VMs to other suitable hosts on a pool. The host must first be disabled using the **host-disable** command.

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

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

### host-forget

`host-forget` uuid=*<Xen Cloud Platform_host_UUID>*

The `xapi` agent forgets about the specified Xen Cloud Platform host without contacting it explicitly.

Use the `--force` parameter to avoid being prompted to confirm that you really want to perform this operation.

**Warning**

Don't use this command if HA is enabled on the pool. Disable HA first, then enable it again after you've forgotten the host.

**Tip**

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

### host-get-system-status

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

Download system status information into the specified file. The optional parameter *`entries`* is a comma-separated list of system status entries, taken from the capabilities XML fragment returned by the **host-get-system-status-capabilities** command. See the section called "host-get-system-status-capabilities" for details. If not specified, all system status information is saved in the file. The parameter *`output`* may be *tar.bz2* (the default) or *zip*; if this parameter is not specified, the file is saved in `tar.bz2` form.

The host(s) on which this operation should be performed are selected using the standard selection mechanism (see host selectors above).

### host-get-system-status-capabilities

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

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

```
<?xml version="1.0" ?> <system-status-capabilities>
   <capability content-type="text/plain" default-checked="yes" key="Xen Cloud Platform-logs"
           max-size="150425200" max-time="-1" min-size="150425200" min-time="-1" \
     pii="maybe"/>
   <capability content-type="text/plain" default-checked="yes" \
     key="Xen Cloud Platform-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 | |

| Attribute | Description |
| --- | --- |
| | 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 | 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:<br>• *no*: no PII will be in these entries<br>• *yes*: PII will likely or certainly be in these entries<br>• *maybe*: you might wish to audit these entries for PII<br>• *if_customized* if the files are unmodified, then they will contain no PII, but since we encourage editing of these files, PII may have been introduced by such customization. This is used in particular for the networking scripts in the control domain.<br>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 using the standard selection mechanism (see host selectors above).

## host-is-in-emergency-mode

`host-is-in-emergency-mode`

Returns *`true`* if the host the CLI is talking to is currently in emergency mode, *`false`* otherwise. This CLI command works directly on slave hosts even with no master host present.

## host-license-add

`host-license-add` license-file=*<path/license_filename>* [host-uuid=*<Xen Cloud Platform_host_UUID>*]

Parses a local license file and adds it to the specified Xen Cloud Platform host.

For details on licensing a host, see ????.

### host-license-view

`host-license-view` [host-uuid=*<Xen Cloud Platform_host_UUID>*]

Displays the contents of the Xen Cloud Platform host license.

### host-logs-download

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

Download a copy of the logs of the specified Xen Cloud Platform 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 using 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. The command should *only* be used from a remote off-host machine where you have space to hold the copy of the logs.

### host-management-disable

`host-management-disable`

Disables the host agent listening on an external management network interface and disconnects all connected API clients. Operates directly on the Xen Cloud Platform host the CLI is connected to, and is not forwarded to the pool master if applied to a member Xen Cloud Platform host.

**Warning**

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

### host-management-reconfigure

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

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

If the UUID of a PIF object is specified, the Xen Cloud Platform host determines 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 that you have network connectivity on the new interface. Use **xe pif-reconfigure** to set one up first. Otherwise, subsequent CLI commands will reach the Xen Cloud Platform host.

### host-reboot

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

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

### host-restore

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

Restore a backup named *`file-name`* of the Xen Cloud Platform 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 from Backup** option.

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

### host-set-hostname-live

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

Change the hostname of the Xen Cloud Platform host specified by *`host-uuid`*. This command persistently sets both the hostname in the control domain database and the actual

Linux hostname of the Xen Cloud Platform host. Note that *hostname* is *not* the same as the value of the *name_label* field.

### host-shutdown

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

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

### host-syslog-reconfigure

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

Reconfigure the `syslog` daemon on the specified Xen Cloud Platform hosts. This command applies the configuration information defined in the host *`logging`* parameter.

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

## Log commands

Commands for working with logs.

### log-get-keys

`log-get-keys`

List the keys of all of the logging subsystems.

### log-reopen

`log-reopen`

Reopen all loggers. Use this command for rotating log files.

### log-set-output

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

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

**xe log-set-output *key=sm level=debug output=<file:/tmp/sm.log>***

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

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

- debug
- info
- warning
- error

## Message commands

Commands for working with messages.

#### Message parameters

| Parameter Name | Description | Type |
|---|---|---|
| uuid | The unique identifier/object reference for the message | read only |
| name | The unique name of the message | read only |
| priority | The message priority. Higher numbers indicate greater priority | read only |
| class | The message class, for example VM. | read only |
| obj-uuid | The uuid of the affected object. | read only |
| timestamp | The time that the message was generated. | read only |
| body | The message content. | read only |

### message-create

`message-create` name=*<message_name>* body=*<message_text>* [[host-uuid=*<uuid_of_host>*] | [sr-uuid=*<uuid_of_sr>*] | [vm-uuid=*<uuid_of_vm>*] | [pool-uuid=*<uuid_of_pool>*]]

Creates a new message.

### message-list

`message-list`

Lists all messages, or messages that match the specified standard selectable parameters.

## Network commands

Commands for working with networks.

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

#### Network parameters

Networks have the following parameters:

| Parameter Name | Description | Type |
|---|---|---|
| uuid | The unique identifier/object reference for the network | read only |
| name-label | The name of the network | read write |
| name-description | The description text of the network | read write |
| 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 Xen Cloud Platform hosts to this network | read only set parameter |
| bridge | name of the bridge corresponding to this network on the local Xen Cloud Platform host | read only |
| other-config:static-routes | comma-separated list of *<subnet>*/*<netmask>*/*<gate-* | read write |

| Parameter Name | Description | Type |
|---|---|---|
| | *way>* formatted entries specifying the gateway address through which to route subnets. For example, setting `other-config:static-routes` to `172.16.0.0/15/192.168.0.3,172.18.0.0/16/192.168.0.4` causes traffic on `172.16.0.0/15` to be routed over `192.168.0.3` and traffic on `172.18.0.0/16` to be routed over `192.168.0.4`. | |
| other-config:ethtool-autoneg | set to `no` to disable autonegotiation of the physical interface or bridge. Default is `yes`. | read write |
| other-config:ethtool-rx | set to `on` to enable receive checksum, `off` to disable | read write |
| other-config:ethtool-tx | set to `on` to enable transmit checksum, `off` to disable | read write |
| other-config:ethtool-sg | set to `on` to enable scatter gather, `off` to disable | read write |
| other-config:ethtool-tso | set to `on` to enable tcp segmentation offload, `off` to disable | read write |
| other-config:ethtool-ufo | set to `on` to enable UDP fragment offload, `off` to disable | read write |
| other-config:ethtool-gso | set to `on` to enable generic segmentation offload, `off` to disable | read write |
| blobs | Binary data store | read only |

## network-create

`network-create` name-label=*<name_for_network>* [name-description=*<descriptive_text>*]

Creates a new network.

## network-destroy

`network-destroy` uuid=*<network_uuid>*

Destroys an existing network.

## Patch (update) commands

Commands for working with Xen Cloud Platform host patches (updates). These are for the standard Xen Cloud Platform.

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

### Patch parameters

Patches have the following parameters:

| Parameter Name | Description | Type |
|---|---|---|
| uuid | The unique identifier/object reference for the patch | read only |
| host-uuid | The unique identifier for the Xen Cloud Platform 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 |

### patch-apply

`patch-apply` uuid=*<patch_file_uuid>*

Apply the specified patch file.

### patch-clean

`patch-clean` uuid=*<patch_file_uuid>*

Delete the specified patch file from the Xen Cloud Platform host.

### patch-pool-apply

`patch-pool-apply` uuid=*<patch_uuid>*

Apply the specified patch to all Xen Cloud Platform hosts in the pool.

### patch-precheck

`patch-precheck` uuid=*<patch_uuid>* host-uuid=*<host_uuid>*

Run the prechecks contained within the specified patch on the specified Xen Cloud Platform host.

### patch-upload

`patch-upload` file-name=*<patch_filename>*

Upload a specified patch file to the Xen Cloud Platform host. This prepares a patch to be applied. On success, the UUID of the uploaded patch is printed out. If the patch has previously been uploaded, a `PATCH_ALREADY_EXISTS` error is returned instead and the patch is not uploaded again.

## PBD commands

Commands for working with PBDs (Physical Block Devices). These are the software objects through which the Xen Cloud Platform host accesses storage repositories (SRs).

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

#### PBD parameters

PBDs have the following parameters:

| 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 SR-backend-driver of a host | read only map parameter |
| currently-attached | True if the SR is currently attached on this host, False otherwise | read only |
| host-uuid | UUID of the physical machine on which the PBD is available | read only |
| host | The host field is deprecated. Use host_uuid instead. | read only |
| other-config | Additional configuration information. | read/write map parameter |

### pbd-create

`pbd-create` host-uuid=*<uuid_of_host>*

sr-uuid=<*uuid_of_sr*>
[device-config:key=<*corresponding_value*>...]

Create a new PBD on a Xen Cloud Platform host. The read-only *`device-config`* parameter can only be set on creation.

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*.

### pbd-destroy

`pbd-destroy` uuid=<*uuid_of_pbd*>

Destroy the specified PBD.

### pbd-plug

`pbd-plug` uuid=<*uuid_of_pbd*>

Attempts to plug in the PBD to the Xen Cloud Platform host. If this succeeds, the referenced SR (and the VDIs contained within) should then become visible to the Xen Cloud Platform host.

### pbd-unplug

`pbd-unplug` uuid=<*uuid_of_pbd*>

Attempt to unplug the PBD from the Xen Cloud Platform host.

## PIF commands

Commands for working with PIFs (objects representing the physical network interfaces).

The PIF objects can be listed with the standard object listing command (**xe pif-list**), and the parameters manipulated with the standard parameter commands. See the section called "Low-level param commands" for details.

#### PIF parameters

PIFs have the following parameters:

| 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 |

| Parameter Name | Description | Type |
|---|---|---|
| MAC | the MAC address of the PIF | read only |
| other-config | Additional PIF configuration name:value pairs. | read/write map parameter |
| 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 |
| 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 Xen Cloud Platform host to which this PIF is connected | read only |
| host-name-label | the name of the Xen Cloud Platform 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| 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 | PCI bus path address | read only |
| other-config:ethtool-speed | sets the speed of connection in Mbps | read write |
| other-config:ethtool-autoneg | set to `no` to disable autonegotiation of the physical interface or bridge. Default is `yes`. | read write |
| other-config:ethtool-duplex | Sets duplexing capability of the PIF, either `full` or `half`. | read write |

| Parameter Name | Description | Type |
| --- | --- | --- |
| other-config:ethtool-rx | set to `on` to enable receive checksum, `off` to disable | read write |
| other-config:ethtool-tx | set to `on` to enable transmit checksum, `off` to disable | read write |
| other-config:ethtool-sg | set to `on` to enable scatter gather, `off` to disable | read write |
| other-config:ethtool-tso | set to `on` to enable tcp segmentation offload, `off` to disable | read write |
| other-config:ethtool-ufo | set to `on` to enable udp fragment offload, `off` to disable | read write |
| other-config:ethtool-gso | set to `on` to enable generic segmentation offload, `off` to disable | read write |
| other-config:domain | comma-separated list used to set the DNS search path | read write |
| other-config:bond-miimon | interval between link liveness checks, in milliseconds | read write |
| other-config:bond-downdelay | number of milliseconds to wait after link is lost before really considering the link to have gone. This allows for transient link loss | read write |
| other-config:bond-updelay | number of milliseconds to wait after the link comes up before really considering it up. Allows for links flapping up. Default is 31s to allow for time for switches to begin forwarding traffic. | read write |
| disallow-unplug | True if this PIF is a dedicated storage NIC, false otherwise | read/write |

**Note**

Changes made to the `other-config` fields of a PIF will only take effect after a reboot. Alternately, use the **xe pif-unplug** and **xe pif-plug** commands to cause the PIF configuration to be rewritten.

### pif-forget

`pif-forget` uuid=*<uuid_of_pif>*

Destroy the specified PIF object on a particular host.

### pif-introduce

`pif-introduce` host-uuid=*<UUID of Xen Cloud Platform 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 Xen Cloud Platform host.

### pif-plug

`pif-plug` uuid=*<uuid_of_pif>*

Attempt to bring up the specified physical interface.

### pif-reconfigure-ip

`pif-reconfigure-ip` uuid=*<uuid_of_pif>* [ mode=*<dhcp>* | mode=*<static>* ] gateway=*<network_gateway_address>* IP=*<static_ip_for_this_pif>* netmask=*<netmask_for_this_pif>* [DNS=*<dns_address>*]

Modify the IP address of the PIF. For static IP configuration, set the *`mode`* parameter to *`static`*, with the *`gateway`*, *`IP`*, and *`netmask`* parameters set to the appropriate values. To use DHCP, set the *`mode`* parameter to *`DHCP`* and leave the static parameters undefined.

### pif-scan

`pif-scan` host-uuid=*<UUID of Xen Cloud Platform host>*

Scan for new physical interfaces on a Xen Cloud Platform host.

### pif-unplug

`pif-unplug` uuid=*<uuid_of_pif>*

Attempt to bring down the specified physical interface.

## Pool commands

Commands for working with pools. A *pool* is an aggregate of one or more Xen Cloud Platform hosts. A pool uses one or more shared storage repositories so that the VMs running on one Xen Cloud Platform 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 Xen Cloud Platform host in the pool. Each Xen Cloud Platform host is really a pool consisting of a single member by default. When a Xen Cloud Platform host is joined to a pool, it is designated as a member, and the pool it has joined becomes the master for the pool.

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

### Pool parameters

Pools have the following parameters:

| 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 Xen Cloud Platform 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 | read/write |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | where suspended VMs on pool members are saved | |
| 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 |
| ha-enabled | True if HA is enabled for the pool, false otherwise | read only |
| ha-configuration | reserved for future use. | read only |
| ha-statefiles | lists the UUIDs of the VDIs being used by HA to determine storage health | read only |
| ha-host-failures-to-tolerate | the number of host failures to tolerate before sending a system alert | read/write |
| ha-plan-exists-for | the number of hosts failures that can actually be handled, according to the calculations of the HA algorithm | read only |
| ha-allow-overcommit | `True` if the pool is allowed to be overcommitted, `False` otherwise | read/write |
| ha-overcommitted | `True` if the pool is currently overcommitted | read only |
| blobs | binary data store | read only |
| wlb-url | Path to the WLB server | read only |
| wlb-username | Name of the user of the WLB service | read only |
| wlb-enabled | True is WLB is enabled | read/write |
| wlb-verify-cert | True if there is a certificate to verify | read/write |

### pool-designate-new-master

`pool-designate-new-master` host-uuid=*<UUID of member Xen Cloud Platform host to become new master>*

Instruct the specified member Xen Cloud Platform host to become the master of an existing pool. This performs an orderly handover of the role of master host to another host in the resource pool. This command only works when the current master is online, and is not a replacement for the emergency mode commands listed below.

### pool-dump-database

`pool-dump-database` file-name=*<filename_to_dump_database_into_(on_client)>*

Download a copy of the entire pool database and dump it into a file on the client.

### pool-eject

`pool-eject` host-uuid=*<UUID of Xen Cloud Platform host to eject>*

Instruct the specified Xen Cloud Platform host to leave an existing pool.

### pool-emergency-reset-master

`pool-emergency-reset-master` master-address=*<address of the pool's master Xen Cloud Platform host>*

Instruct a slave member Xen Cloud Platform host to reset its master address to the new value and attempt to connect to it. This command should not be run on master hosts.

### pool-emergency-transition-to-master

`pool-emergency-transition-to-master`

Instruct a member Xen Cloud Platform host to become the pool master. This command is only accepted by the Xen Cloud Platform host if it has transitioned to emergency mode, meaning it is a member of a pool whose master has disappeared from the network and could not be contacted for some number of retries.

Note that this command may cause the password of the host to reset if it has been modified since joining the pool (see the section called “User commands”).

### pool-join

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

Instruct a Xen Cloud Platform host to join an existing pool.

### pool-recover-slaves

`pool-recover-slaves`

Instruct the pool master to try and reset the master address of all members currently running in emergency mode. This is typically used after **pool-emergency-transition-to-master** has been used to set one of the members as the new master.

### pool-restore-database

`pool-restore-database` file-name=*<filename_to_restore_from_(on_client)>* [dry-run=*<true | false>*]

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.

There is also a *dry run* option, which allows you to check that the pool database can be restored without actually perform the operation. By default, `dry-run` is set to false.

### 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 automatically replicated, but can be useful for ensuring changes are rapidly replicated after performing a significant set of CLI operations.

## Storage Manager commands

Commands for controlling Storage Manager plugins.

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

#### SM parameters

SMs have the following parameters:

| 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 |

| Parameter Name | Description | Type |
|---|---|---|
| copyright | copyright statement for this SM plugin | read only |
| required-api-version | minimum SM API version required on the Xen Cloud Platform host | read only |
| configuration | names and descriptions of device configuration keys | read only |
| capabilities | capabilities of the SM plugin | read only |
| driver-filename | the filename of the SR driver. | read only |

## SR commands

Commands for controlling SRs (storage repositories).

The SR objects can be listed with the standard object listing command (**xe sr-list**), and the parameters manipulated with the standard parameter commands. See the section called "Low-level param commands" for details.

### SR parameters

SRs have the following parameters:

| 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 | list of the operations that are currently in progress on this SR | read only set parameter |
| VDIs | unique identifier/object reference for the virtual disks in this SR | read only set parameter |
| PBDs | unique identifier/object reference for the PBDs attached to this SR | read only set parameter |
| physical-utilisation | physical space currently utilized on this SR, in bytes. Note that for sparse disk formats, physical uti- | read only |

| Parameter Name | Description | Type |
|---|---|---|
| | lization may be less than virtual allocation | |
| physical-size | total physical size of the SR, in bytes | read only |
| type | type of the SR, used to specify the SR backend driver to use | read only |
| content-type | the type of the SR's content. Used 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, Xen.org recommends 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 Xen Cloud Platform hosts; False otherwise | read/write |
| other-config | 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-size values of all VDIs in this storage repository (in bytes) | read only |
| sm-config | SM dependent data | read only map parameter |
| blobs | binary data store | read only |

### sr-create

sr-create name-label=<*name*> physical-size=<*size*> type=<*type*> content-type=<*content_type*> device-config:<*config_name*>=<*value*> [host-uuid=<*Xen Cloud Platform host UUID*>] [shared=<*true | false*>]

Creates an SR on the disk, introduces it into the database, and creates a PBD attaching the SR to a Xen Cloud Platform host. If *shared* is set to *true*, a PBD is created for each Xen Cloud Platform host in the pool; if *shared* is not specified or set to *false*, a PBD is created only for the Xen Cloud Platform 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.

### sr-destroy

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

Destroys the specified SR on the Xen Cloud Platform host.

### sr-forget

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

The `xapi` agent forgets about a specified SR on the Xen Cloud Platform host, meaning that the SR is detached and you cannot access VDIs on it, but it remains intact on the source media (the data is not lost).

### sr-introduce

`sr-introduce` name-label=*<name>*
physical-size=*<physical_size>*
type=*<type>*
content-type=*<content_type>*
uuid=*<sr_uuid>*

Just places an SR record into the database. The `device-config` parameters are specified by `device-config:`*`<parameter_key>`*=*`<parameter_value>`*, for example:

```
xe sr-introduce 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 it was created, or to help recover from various failure scenarios.

### sr-probe

`sr-probe` type=*<type>* [host-uuid=*<uuid_of_host>*] [device-config:*<config_name>*=*<value>*]

Performs a backend-specific scan, using the provided *`device-config`* keys. If the *`device-config`* is complete for the SR backend, then this will return a list of the SRs present on the device, if any. If the *`device-config`* parameters are only partial, then a backend-specific scan will be performed, returning results that will guide you in improving the remaining *`device-config`* parameters. The scan results are returned as backend-specific XML, printed out on 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.

### sr-scan

`sr-scan` uuid=*<sr_uuid>*

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

## Task commands

Commands for working with long-running asynchronous tasks. These are tasks such as starting, stopping, and suspending a Virtual Machine, which are typically made up of a set of other atomic subtasks that together accomplish the requested operation.

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

#### Task parameters

Tasks have the following parameters:

| Parameter Name | Description | Type |
|---|---|---|
| uuid | the unique identifier/object reference for the Task | read only |
| name-label | the name of the Task | read only |
| 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 successfully completed, 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| 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 |
| subtask_of | contains the UUID of the tasks this task is a sub-task of | read only |
| subtasks | contains the UUID(s) of all the subtasks of this task | read only |

### task-cancel

`task-cancel` [uuid=*<task_uuid>*]

Direct the specified Task to cancel and return.

## Template commands

Commands for working with VM templates.

Templates are essentially VMs with the *`is-a-template`* parameter set to *`true`*. A template is a "gold image" that contains all the various configuration settings to instantiate a specific VM. Xen Cloud Platform 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). With Xen Cloud Platform you can create VMs, configure them in standard forms for your particular needs, and save a copy of them as templates for future use in VM deployment.

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

### Template parameters

Templates have the following parameters:

| 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<br><br>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* | read/write |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | >= *memory_static_min*. | |
| 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.<br><br>You can tune a VCPU's pinning with<br><br>`xe vm-param-set \ uuid=<vm_uuid> \ VCPUs-params:mask=1,2,3`<br><br>A VM created from this template will then run on | read/write map parameter |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | physical CPUs 1, 2, and 3 only.<br><br>You can also tune the VCPU priority (xen scheduling) with the *cap* and *weight* parameters; for example<br><br>`xe vm-param-set \ uuid=<vm_uuid> \ VCPUs-params:weight=512 xe vm-param-set \ uuid=<vm_uuid> \ VCPUs-params:cap=100`<br><br>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 Xen Cloud Platform host. Legal weights range from 1 to 65535 and the default is 256.<br><br>The cap optionally fixes the maximum amount of CPU a VM based on this template will be able to consume, even if the Xen Cloud Platform 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| 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 | 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 |
| 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 | the boot policy for HVM guests. Either `BIOS Order` or an empty string. | 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| 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 Xen Cloud Platform host on which a VM based on this template is currently resident; appears as `<not in database>` for a template | read only |
| affinity | a Xen Cloud Platform 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 | 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 `yyyymmddThh:mm:ss` | read only |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | `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 | |
| install-time | timestamp of the date and time that the metrics for a VM based on this template were read, in the form `yyyymmddThh:mm:ss z`, where `z` is the single-letter military timezone indicator, for example, Z for UTC (GMT); set to 1 Jan 1970 Z (beginning of Unix/POSIX epoch) for a template | read only |
| 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 | list of virtual CPUs and their weight | read only map parameter |
| 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| 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 yyyymmddThh:mm:ss z, where z is the single-letter military time-zone indicator, for example, Z for UTC (GMT) | read only |
| actions-after-shutdown | action to take after the VM has shutdown | read/write |

| Parameter Name | Description | Type |
|---|---|---|
| actions-after-reboot | action to take after the VM has rebooted | read/write |
| 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 | 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 |
| is-a-snapshot | True if this template is a VM snapshot | read only |
| snapshot_of | the UUID of the VM that this template is a snapshot of | read only |
| snapshots | the UUID(s) of any snapshots that have been taken of this template | read only |
| snapshot_time | the timestamp of the most recent VM snapshot taken | read only |
| memory-target | the target amount of memory set for this template | read only |
| blocked-operations | lists the operations that cannot be performed on this template | read/write map parameter |

| Parameter Name | Description | Type |
| --- | --- | --- |
| last-boot-record | record of the last boot parameters for this template, in XML format | read only |
| ha-always-run | `True` if an instance of this template will always restarted on another host in case of the failure of the host it is resident on | read/write |
| ha-restart-priority | 1, 2, 3 or best effort. 1 is the highest restart priority | read/write |
| blobs | binary data store | read only |
| live | only relevant to a running VM. | read only |

### 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.

## User commands

### user-password-change

`user-password-change` old=*<old_password>* new=*<new_password>*

Changes the password of the logged-in user. The old password field is not checked because you require supervisor privilege to make this call.

## VBD commands

Commands for working with VBDs (Virtual Block Devices).

A VBD is a software object that connects a VM to the VDI, which represents the contents of the virtual disk. The VBD has the attributes which tie the VDI to the VM (is it bootable, its read/write metrics, and so on), while the VDI has the information on the physical attributes of the virtual disk (which type of SR, whether the disk is shareable, whether the media is read/write or read only, and so on).

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

### VBD parameters

VBDs have the following parameters:

| 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 | `True` if the VBD is currently attached on this host, `false` otherwise | read only |
| storage-lock | `True` if a storage-level lock was acquired | read only |
| status-code | | read only |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | error/success code associated with the last attach operation | |
| 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 per second for this VBD | read only |
| io_write_kbs | average write rate in kB per second for this VBD | 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 | 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 |
| 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 |

### vbd-create

`vbd-create` vm-uuid=*<uuid_of_the_vm>* device=*<device_value>* vdi-uuid=*<uuid_of_the_vdi_the_vbd_will_connect_to>* [bootable=true] [type=*<Disk | CD>*] [mode=*<RW | RO>*]

Create a new VBD on a VM.

Appropriate values for the *`device`* field are listed in the parameter *`allowed-VBD-devices`* on the specified VM. Before any VBDs exist there, the allowable values are integers from 0-15.

If the *`type`* is *`Disk`*, *`vdi-uuid`* is required. Mode can be *`RO`* or *`RW`* for a Disk.

If the *`type`* is *`CD`*, *`vdi-uuid`* is optional; if no VDI is specified, an empty VBD will be created for the CD. Mode must be *`RO`* for a CD.

### vbd-destroy

`vbd-destroy` uuid=*<uuid_of_vbd>*

Destroy the specified VBD.

If the VBD has its *`other-config:owner`* parameter set to *`true`*, the associated VDI will also be destroyed.

### vbd-eject

`vbd-eject` uuid=*<uuid_of_vbd>*

Remove the media from the drive represented by a VBD. This command only works if the media is of a removable type (a physical CD or an ISO); otherwise an error message `VBD_NOT_REMOVABLE_MEDIA` is returned.

### vbd-insert

`vbd-insert` uuid=*<uuid_of_vbd>* vdi-uuid=*<uuid_of_vdi_containing_media>*

Insert new media into the drive represented by a VBD. This command only works if the media is of a removable type (a physical CD or an ISO); otherwise an error message `VBD_NOT_REMOVABLE_MEDIA` is returned.

### vbd-plug

`vbd-plug` uuid=*<uuid_of_vbd>*

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

### vbd-unplug

`vbd-unplug` uuid=*<uuid_of_vbd>*

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

## VDI commands

Commands for working with VDIs (Virtual Disk Images).

A VDI is a software object that represents the contents of the virtual disk seen by a VM, as opposed to the VBD, which is a connector object that ties a VM to the VDI. The VDI has the information on the physical attributes of the virtual disk (which type of SR, whether the disk is shareable, whether the media is read/write or read only, and so on), while the VBD has the attributes which tie the VDI to the VM (is it bootable, its read/write metrics, and so on).

The VDI objects can be listed with the standard object listing command (**xe vdi-list**), and the parameters manipulated with the standard parameter commands. See the section called "Low-level param commands" for details.

### VDI parameters

VDIs have the following parameters:

| 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 the 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 |

| Parameter Name | Description | Type |
|---|---|---|
| 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 |
| location | location information | read only |
| managed | true if the VDI 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 |
| is-a-snapshot | true if this VDI is a VM storage snapshot | read only |
| snapshot_of | the UUID of the storage this VDI is a snapshot of | read only |
| snapshots | the UUID(s) of all snapshots of this VDI | read only |
| snapshot_time | the timestamp of the snapshot operation that created this VDI | read only |

### vdi-clone

`vdi-clone` uuid=*<uuid_of_the_vdi>* [driver-params:*<key=value>*]

Create 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 driver documentation for details.

### vdi-copy

`vdi-copy` uuid=*<uuid_of_the_vdi>* sr-uuid=*<uuid_of_the_destination_sr>*

Copy a VDI to a specified SR.

### vdi-create

`vdi-create` sr-uuid=*<uuid_of_the_sr_where_you_want_to_create_the_vdi>*
name-label=*<name_for_the_vdi>*
type=*<system | user | suspend | crashdump>*
virtual-size=*<size_of_virtual_disk>*
sm-config-*=*<storage_specific_configuration_data>*

Create a VDI.

The *virtual-size* parameter can be specified in bytes or using the IEC standard suffixes KiB ($2^{10}$ bytes), MiB ($2^{20}$ bytes), GiB ($2^{30}$ bytes), and TiB ($2^{40}$ bytes).

**Note**

SR types that support sparse allocation of disks (such as Local VHD and NFS) do not enforce virtual allocation of disks. Users should therefore take great care when over-allocating virtual disk space on an SR. If an over-allocated SR does become full, disk space must be made available either on the SR target substrate or by deleting unused VDIs in the SR.

**Note**

Some SR types might round up the *virtual-size* value to make it divisible by a configured block size.

### vdi-destroy

`vdi-destroy` uuid=*<uuid_of_vdi>*

Destroy the specified VDI.

**Note**

In the case of Local VHD and NFS SR types, disk space is not immediately released on **vdi-destroy**, but periodically during a storage repository scan operation. Users that need to force deleted disk space to be made available should call **sr-scan** manually.

### vdi-forget

`vdi-forget` uuid=*<uuid_of_vdi>*

Unconditionally removes a VDI record from the database without touching the storage back-end. In normal operation, you should be using **vdi-destroy** instead.

### vdi-import

`vdi-import` uuid=*<uuid_of_vdi>* filename=*<filename_of_raw_vdi>*

Import a raw VDI.

### vdi-introduce

`vdi-introduce` uuid=*<uuid_of_vdi>*
sr-uuid=*<uuid_of_sr_to_import_into>*
name-label=*<name_of_the_new_vdi>*
type=*<system | user | suspend | crashdump>*
location=*<device_location_(varies_by_storage_type)>*
[name-description=*<description_of_vdi>*]
[sharable=*<yes | no>*]
[read-only=*<yes | no>*]
[other-config=*<map_to_store_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.

### vdi-resize

`vdi-resize` uuid=*<vdi_uuid>* disk-size=*<new_size_for_disk>*

Resize the VDI specified by UUID.

### vdi-snapshot

`vdi-snapshot` uuid=*<uuid_of_the_vdi>* [driver-params=*<params>*]

Produces a read-write version of a VDI that can be used as a reference for backup and/or templating purposes. You can perform a backup from a snapshot rather than installing and running backup software inside the VM. The VM can continue running while external backup software streams the contents of the snapshot to the backup media. Similarly, a snapshot can be used as a "gold image" on which to base a template. A template can be made using any VDIs.

The optional *`driver-params`* map parameter can be used for passing extra vendor-specific configuration information to the back end storage driver that the VDI is based on. See the storage vendor driver documentation for details.

A clone of a snapshot should always produce a writable VDI.

### vdi-unlock

`vdi-unlock` uuid=*<uuid_of_vdi_to_unlock>* [force=true]

Attempts to unlock the specified VDIs. If *`force=true`* is passed to the command, it will force the unlocking operation.

## VIF commands

Commands for working with VIFs (Virtual network interfaces).

The VIF objects can be listed with the standard object listing command (**xe vif-list**), and the parameters manipulated with the standard parameter commands. See the section called "Low-level param commands" for details.

### VIF parameters

VIFs have the following parameters:

| 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 using the other-config *map* parameter. For example, to reset the MTU on a virtual NIC to use jumbo frames: | read only |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | `xe vif-param-set \ uuid=<vif_uuid> \ other-config:mtu=9000` | |
| 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 |
| MAC-autogenerated | True if the MAC address of the VIF was automatically generated | read only |
| other-config | Additional configuration key:value pairs | read/write map parameter |
| other-config:ethtool-rx | set to `on` to enable receive checksum, `off` to disable | read write |
| other-config:ethtool-tx | set to `on` to enable transmit checksum, `off` to disable | read write |
| other-config:ethtool-sg | set to `on` to enable scatter gather, `off` to disable | read write |
| other-config:ethtool-tso | set to `on` to enable tcp segmentation offload, `off` to disable | read write |
| other-config:ethtool-ufo | set to `on` to enable udp fragment offload, `off` to disable | read write |
| other-config:ethtool-gso | set to `on` to enable generic segmentation offload, `off` to disable | read write |
| other-config:promiscuous | `true` to a VIF to be promiscuous on the bridge, so that it sees all | read write |

| Parameter Name | Description | Type |
|---|---|---|
| | traffic over the bridge. Useful for running an Intrusion Detection System (IDS) or similar in a VM. | |
| 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-create

`vif-create` vm-uuid=*<uuid_of_the_vm>* device=*<see below>* network-uuid=*<uuid_of_the_network_the_vif_will_connect_to>* [mac=*<mac_address>*]

Create a new VIF on a VM.

Appropriate values for the *`device`* field are listed in the parameter *`allowed-VIF-devices`* on the specified VM. Before any VIFs exist there, the allowable values are integers from `0-15`.

The *`mac`* parameter is the standard MAC address in the form `aa:bb:cc:dd:ee:ff`. If you leave it unspecified, an appropriate random MAC address will be created. You can also explicitly set a random MAC address by specifying *`mac=random`*.

### vif-destroy

`vif-destroy` uuid=*<uuid_of_vif>*

Destroy a VIF.

### vif-plug

`vif-plug` uuid=*<uuid_of_vif>*

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

### vif-unplug

`vif-unplug` uuid=*<uuid_of_vif>*

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

## VLAN commands

Commands for working with VLANs (virtual networks). To list and edit virtual interfaces, refer to the PIF commands, which have a VLAN parameter to signal that they have an associated virtual network (see the section called “PIF commands”). For example, to list VLANs you need to use **xe pif-list**.

### vlan-create

`vlan-create` pif-uuid=*<uuid_of_pif>* vlan=*<vlan_number>* network-uuid=*<uuid_of_network>*

Create a new VLAN on a Xen Cloud Platform host.

### pool-vlan-create

`vlan-create` pif-uuid=*<uuid_of_pif>* vlan=*<vlan_number>* network-uuid=*<uuid_of_network>*

Create a new VLAN on all hosts on a pool, by determining which interface (for example, `eth0`) the specified network is on (on each host) and creating and plugging a new PIF object one each host accordingly.

### vlan-destroy

`vlan-destroy` uuid=*<uuid_of_pif_mapped_to_vlan>*

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

## VM commands

Commands for controlling VMs and their attributes.

#### VM selectors

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

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

#### VM parameters

VMs have the following parameters:

**Note**

All writeable VM parameter values can be changed while the VM is running, but the new parameters are *not* applied dynamically and will not be applied until the VM is rebooted.

| 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<br><br>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.<br><br>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 val- | read/write |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | ue, the VM must be shut down. | |
| suspend-VDI-uuid | the VDI that a suspend image is stored on | read only |
| VCPUs-params | configuration parameters for the selected VCPU policy.<br><br>You can tune a VCPU's pinning with<br><br>`xe vm-param-set \ uuid=<vm_uuid> \ VCPUs-params:mask=1,2,3`<br><br>The selected VM will then run on physical CPUs 1, 2, and 3 only.<br><br>You can also tune the VCPU priority (xen scheduling) with the *cap* and *weight* parameters; for example<br><br>`xe vm-param-set \ uuid=<template_uuid> \ VCPUs-params:weight=512 xe vm-param-set \ uuid=<template UUID> \ VCPUs-params:cap=100`<br><br>A VM with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended Xen Cloud Platform host. Legal weights range from 1 to 65535 and the default is 256.<br><br>The cap optionally fixes the maximum amount of CPU a VM will be able to consume, even if the Xen Cloud Plat- | read/write map parameter |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | form 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 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 destroy (no coredump and leave VM halted). | 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 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 | read only set parameter |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | other devices may be used (but may not work). | |
| 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 | the boot policy for HVM guests. Either `BIOS Order` or an empty string. | 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 |
| 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| PV-bootloader-args | string of miscellaneous arguments for the boot-loader | read/write |
| last-boot-CPU-flags | describes the CPU flags on which the VM was last booted | read only |
| resident-on | the Xen Cloud Platform host on which a VM is currently resident | read only |
| affinity | a Xen Cloud Platform 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<br><br>For example, a VM will be started automatically after host boot if the other-config parameter includes the key/value pair *auto_poweron: true* | read/write map parameter |
| start-time | timestamp of the date and time that the metrics for the VM were read, in the form `yyyymmddThh:mm:ss z`, where `z` is the single-letter military time-zone 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 `yyyymmddThh:mm:ss z`, where `z` is the single-letter military time- | read only |

| Parameter Name | Description | Type |
| --- | --- | --- |
| | zone indicator, for example, Z for UTC (GMT) | |
| memory-actual | the actual memory being used by a VM | read only |
| VCPUs-number | the number of virtual CPUs assigned to the VM<br><br>For a paravirtualized Linux VM, this number can differ from *VCPUS-max* and can be changed without rebooting the VM using the **vm-vcpu-hotplug** command. See the section called “vm-vcpu-hotplug”. Windows VMs always run with the number of vCPUs set to *VCPUs-max* and must be rebooted to change this value.<br><br>Note that performance will drop sharply if you set *VCPUs-number* to a value greater than the number of physical CPUs on the Xen Cloud Platform 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 |

| Parameter Name | Description | Type |
| --- | --- | --- |
| 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 `yyyymmddThh:mm:ss z`, where `z` is the single-letter military time-zone 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 |
| possible-hosts | potential hosts of this VM | read only |
| dom-id | domain ID (if available, -1 otherwise) | read only |
| recommendations | 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 |
| is-a-snapshot | `True` if this VM is a snapshot | read only |
| snapshot_of | the UUID of the VM this is a snapshot of | read only |

| Parameter Name | Description | Type |
|---|---|---|
| snapshots | the UUID(s) of all snapshots of this VM | read only |
| snapshot_time | the timestamp of the snapshot operation that created this VM snapshot | read only |
| memory-target | the target amount of memory set for this VM | read only |
| blocked-operations | lists the operations that cannot be performed on this VM | read/write map parameter |
| last-boot-record | record of the last boot parameters for this template, in XML format | read only |
| ha-always-run | `True` if this VM will always restarted on another host in case of the failure of the host it is resident on | read/write |
| ha-restart-priority | 1, 2, 3 or best effort. 1 is the highest restart priority | read/write |
| blobs | binary data store | read only |
| live | `True` if the VM is running, false if HA suspects that the VM may not be running. | read only |

## 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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-cd-eject

`vm-cd-eject` [*<vm-selector>*=*<vm_selector_value>*...]

Eject a CD from the virtual CD drive. This command only works if there is one and only one CD attached to the VM. When there are two or more CDs, 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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-cd-insert

`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, use the **xe vbd-insert** command 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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-cd-list

`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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

You can also select which VBD and VDI parameters to list.

### vm-cd-remove

`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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-clone

`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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-compute-maximum-memory

vm-compute-maximum-memory total=<*amount_of_available_physical_ram_in_bytes*> [approximate=<*add overhead memory for additional vCPUS? true | false*>] [<*vm_selector*>=<*vm_selector_value*>...]

Calculate the maximum amount of static memory which can be allocated to an existing VM, using the total amount of physical RAM as an upper bound. The optional parameter *approximate* reserves sufficient extra memory in the calculation to account for adding extra vCPUs into the VM at a later date.

For example:

```
xe vm-compute-maximum-memory vm=testvm total=`xe host-list params=memory-free --minimal`
```

This command 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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-copy

vm-copy new-name-label=<*name_for_copy*> [new-name-description=<*description_for_copy*>] [sr-uuid=<*uuid_of_sr*>] [<*vm-selector*>=<*vm_selector_value*>...]

Copy an existing VM, but without using storage-level fast disk clone operation (even if this is available). The disk images of the copied VM are guaranteed to be "full images" - that is, not part of a copy-on-write (CoW) chain.

Specify the name and the optional description for the resulting copied VM using the *new-name-label* and *new-name-description* arguments.

Specify the destination SR for the resulting copied VM using the *sr-uuid*. If this parameter is not specified, the destination is the same SR that the original VM is in.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-crashdump-list

vm-crashdump-list [<*vm-selector*>=<*vm selector value*>...]

List crashdumps associated with the specified VMs.

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

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-data-source-forget

`vm-data-source-forget` data-source=*<name_description_of_data-source>* [*<vm-selector>*=*<vm selector value>*...]

Stop recording the specified data source for a VM, and forget all of the recorded data.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-data-source-list

`vm-data-source-list` [*<vm-selector>*=*<vm selector value>*...]

List the data sources that can be recorded for a VM.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-data-source-query

`vm-data-source-query` data-source=*<name_description_of_data-source>* [*<vm-selector>*=*<vm selector value>*...]

Display the specified data source for a VM.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-data-source-record

`vm-data-source-record` data-source=*<name_description_of_data-source>* [*<vm-selector>*=*<vm selector value>*...]

Record the specified data source for a VM.

This will write the information from the data source to the VM's persistent performance metrics database. This database is distinct from the normal agent database for performance reasons.

Data sources have the true/false parameters *standard* and *enabled*, which can be seen in the output of the **vm-data-source-list** command. If *enabled=true*, the data source metrics are currently being recorded to the performance database; if *enabled=false* they are not. Data sources with *standard=true* have *enabled=true* and have their metrics recorded to the performance database by default. Data sources which have *standard=false* have *enabled=false* by default. The **vm-data-source-record** command sets *enabled=false*.

Once enabled, you can stop recording the metrics of the data source by running the **vm-data-source-forget** command.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-destroy

`vm-destroy` uuid=*<uuid_of_vm>*

Destroy the specified VM. This leaves the storage associated with the VM intact. To delete storage as well, use **xe vm-uninstall**.

### vm-disk-add

`vm-disk-add` disk-size=*<size_of_disk_to_add>* device=*<uuid_of_device>* [*<vm-selector>*=*<vm_selector_value>*...]

Add a new disk to the specified VMs. Select the *device* parameter 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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-disk-list

`vm-disk-list` [vbd-params] [vdi-params] [*<vm-selector>*=*<vm_selector_value>*...]

Lists disks attached to the specified VMs. The *vbd-params* and *vdi-params* parameters control the fields of the respective objects to output and should be given as a comma-separated list, or the special key *all* for the complete list.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-disk-remove

`vm-disk-remove` device=*<integer_label_of_disk>* [*<vm-selector>*=*<vm_selector_value>*...]

Remove a disk from the specified VMs and destroy it.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-export

`vm-export` filename=*<export_filename>*
[metadata=*<true | false>*]
[*<vm-selector>*=*<vm_selector_value>*...]

Export the specified VMs (including disk images) to a file on the local machine. Specify the filename to export the VM into using the *`filename`* parameter. By convention, the filename should have a `.xva` extension.

If the *`metadata`* parameter is *`true`*, then the disks are not exported, and only the VM metadata is written to the output file. This is intended to be used when the underlying storage is transferred through other mechanisms, and permits the VM information to be recreated (see the section called "vm-import").

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-import

`vm-import` filename=*<export_filename>*
[metadata=*<true | false>*]
[preserve=*<true | false>*]
[sr-uuid=*<destination_sr_uuid>*]

Import a VM from a previously-exported file. If *`preserve`* is set to *`true`*, the MAC address of the original VM will be preserved. The *`sr-uuid`* determines the destination SR to import the VM into, and is the default SR if not specified.

The *`filename`* parameter can also point to an XVA-format VM, which is the legacy export format from Xen Cloud Platform 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 or 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).

**Note**

Multiple VM imports will be performed faster in serial that in parallel.

### vm-install

`vm-install` new-name-label=*<name>*
[ template-uuid=*<uuid_of_desired_template>* |
[template=*<uuid_or_name_of_desired_template>*]]
[ sr-uuid=*<sr_uuid>* | sr-name-label=*<name_of_sr>* ]

Install a VM from a template. Specify the template name using either the *`template-uuid`* or *`template`* argument. Specify an SR other than the default SR using either the *`sr-uuid`* or *`sr-name-label`* argument.

### vm-memory-shadow-multiplier-set

`vm-memory-shadow-multiplier-set` [*<vm-selector>*=*<vm_selector_value>*...]
[multiplier=*<float_memory_multiplier>*]

Set the shadow memory multiplier for the specified VM.

This is an advanced option which modifies the amount of *shadow memory* assigned to a hardware-assisted VM. In some specialized application workloads, such as Xen.org XenApp, extra shadow memory is required to achieve full performance.

This memory is considered to be an overhead. It is separated from the normal memory calculations for accounting memory to a VM. When this command is invoked, the amount of free Xen Cloud Platform 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 Xen Cloud Platform host memory free, then an error will be returned.

The VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors for more information).

### vm-migrate

`vm-migrate` [[host-uuid=*<destination Xen Cloud Platform host UUID>* ] | [host=*<name or UUID of destination Xen Cloud Platform 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 Xen Cloud Platform 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 ex-

tremely 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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-reboot

`vm-reboot` [*<vm-selector>*=*<vm_selector_value>*...] [force=*<true>*]

Reboot the specified VMs.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

Use the *`force`* argument to cause an ungraceful shutdown, akin to pulling the plug on a physical server.

### vm-reset-powerstate

`vm-reset-powerstate` [*<vm-selector>*=*<vm_selector_value>*...] {force=true}

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

This is an *advanced* command only to be used when a member host in a pool goes down. You can use this command to force the pool master to reset the power-state of the VMs to be `halted`. Essentially this forces the lock on the VM and its disks so it can be subsequently started on another pool host. This call *requires* the force flag to be specified, and fails if it is not on the command-line.

### vm-resume

`vm-resume` [*<vm-selector>*=*<vm_selector_value>*...] [force=*<true | false>*] [on=*<Xen Cloud Platform host UUID>*]

Resume the specified VMs.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

If the VM is on a shared SR in a pool of hosts, use the *on* argument to specify which host in the pool on which to start it. By default the system will determine an appropriate host, which might be any of the members of the pool.

### vm-shutdown

`vm-shutdown` [*<vm-selector>*=*<vm_selector_value>*...] [force=*<true | false>*]

Shut down the specified VM.

The VM or VMs on which this operation should be performed are selected using 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, similar to pulling the plug on a physical server.

### vm-start

`vm-start` [*<vm-selector>*=*<vm_selector_value>*...] [force=*<true | false>*] [on=*<Xen Cloud Platform host UUID>*] [--multiple]

Start the specified VMs.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

If the VMs are on a shared SR in a pool of hosts, use the *on* argument to specify which host in the pool on which to start the VMs. By default the system will determine an appropriate host, which might be any of the members of the pool.

### vm-suspend

`vm-suspend` [*<vm-selector>*=*<vm_selector_value>*...]

Suspend the specified VM.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-uninstall

`vm-uninstall` [*<vm-selector>*=*<vm_selector_value>*...] [force=*<true | false>*]

Uninstall a VM, destroying its disks (those VDIs that are marked RW and connected to this VM only) as well as its metadata record. To simply destroy the VM metadata, use **xe vm-destroy**.

The VM or VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-vcpu-hotplug

`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 using the standard selection mechanism (see VM selectors). Optional arguments can be any number of the VM parameters listed at the beginning of this section.

### vm-vif-list

`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 using 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.

## Workload Balancing commands

Commands for controlling the Workload Balancing feature.

### pool-initialize-wlb

```
pool-initialize-wlb wlb_url=<wlb_server_address> \
wlb_username=<wlb_server_username> \
wlb_password=<wlb_server_password> \
Xen Cloud Platform_username=<pool_master_username> \
Xen Cloud Platform_password=<pool_master_password>
```

Starts the workload balancing service on a pool.

### pool-param-set other-config

Use the **pool-param-set other-config** command to specify the timeout when communicating with the WLB server. All requests are serialized, and the timeout covers the time from a request being queued to its response being completed. In other words, slow calls cause subsequent ones to be slow. Defaults to 30 seconds if unspecified or if the setting cannot be parsed.

```
xe pool-param-set other-config:wlb_timeout=<0.01> \
uuid=<315688af-5741-cc4d-9046-3b9cea716f69>
```

### host-retrieve-wlb-evacuate-recommendations

```
host-retrieve-wlb-evacuate-recommendations uuid=<vm_uuid>
```

Returns the evacuation recommendations for a host, and a reference to the UUID of the recommendations object.

### vm-retrieve-wlb-recommendations

Returns the workload balancing recommendations for the selected VM. The simplest way to select the VM on which the operation is to be performed 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` selects all VMs whose power-state is halted. Where multiple VMs are matching, specify the option `--multiple` to perform the operation. The full list of fields that can be matched 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.

### pool-deconfigure-wlb

Permanently deletes all workload balancing configuration.

### pool-retrieve-wlb-configuration

Prints all workload balancing configuration to standard out.

### pool-retrieve-wlb-recommendations

Prints all workload balancing recommendations to standard out.

### pool-retrieve-wlb-report

Gets a WLB report of the specified type and saves it to the specified file. The available reports are:

- `pool_health`
- `host_health_history`
- `optimization_performance_history`
- `pool_health_history`
- `vm_movement_history`
- `vm_performance_history`

Example usage for each report type is shown below. The *`utcoffset`* parameter specifies the number of hours ahead or behind of UTC for your time zone. The *`start`* parameter and *end* parameters specify the number of hours to report about. For example specifying *`start=-3`* and *`end=0`* will cause WLB to report on the last 3 hour of activity.

```
xe pool-retrieve-wlb-report report=pool_health \
poolid=<51e411f1-62f4-e462-f1ed-97c626703cae> \
utcoffset=<-5> \
start=<-3> \
end=<0> \
filename=</pool_health.txt>
```

```
xe pool-retrieve-wlb-report report=host_health_history \
hostid=<e26685cd-1789-4f90-8e47-a4fd0509b4a4> \
utcoffset=<-5> \
start=<-3> \
end=<0> \
filename=</host_health_history.txt>
```

```
xe pool-retrieve-wlb-report report=optimization_performance_history \
poolid=<51e411f1-62f4-e462-f1ed-97c626703cae> \
utcoffset=<-5> \
start=<-3> \
end=<0> \
filename=</optimization_performance_history.txt>
```

```
xe pool-retrieve-wlb-report report=pool_health_history \
poolid=<51e411f1-62f4-e462-f1ed-97c626703cae> \
utcoffset=<-5> \
start=<-3> \
end=<0> \
<filename=/pool_health_history.txt>
```

```
xe pool-retrieve-wlb-report report=vm_movement_history \
poolid=<51e411f1-62f4-e462-f1ed-97c626703cae> \
utcoffset=<-5> \
start=<-5> \
end=<0> \
filename=</vm_movement_history.txt>
```

```
xe pool-retrieve-wlb-report report=vm_performance_history \
hostid=<e26685cd-1789-4f90-8e47-a4fd0509b4a4> \
utcoffset=<-5> \
start=<-3> \
end=<0> \
<filename=/vm_performance_history.txt>
```

# Chapter 9. Troubleshooting

If you experience odd behavior, application crashes, or have other issues with a Xen Cloud Platform 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 you track and resolve the issue.

Troubleshooting of installation issues is covered in the *Xen Cloud Platform Installation Guide*. Troubleshooting of Virtual Machine issues is covered in the *Xen Cloud Platform Virtual Machine Installation Guide*.

Xen.org provides support at Support site.

## Xen Cloud Platform host logs

The Xen Cloud Platform 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 Xen.org Support ftp site. Please refer to the section called “host-bugreport-upload” for a full description of this command and its optional parameters. If you are requested to send a crashdump to Xen.org Support, use the xe command **host-crashdump-upload**. Please refer to the section called “host-crashdump-upload” for a full description of this command and its optional parameters.

### Caution

It is possible that sensitive information might be written into the Xen Cloud Platform host logs.

By default, the server logs report only errors and warnings. If you need to see more detailed information, you can enable more verbose logging. To do so, use the **host-loglevel-set** command:

**host-loglevel-set** *log-level=level*

where *level* can be 0, 1, 2, 3, or 4, where 0 is the most verbose and 4 is the least verbose.

Log files greater than 5 MB are rotated, keeping 4 revisions. The **logrotate** command is run hourly.

## Sending host log messages to a central server

Rather than have logs written to the control domain filesystem, you can configure a Xen Cloud Platform host to write them to a remote server. The remote server must have the syslogd daemon running on it to receive the logs and aggregate them correctly. The syslogd daemon is a standard part of all flavors of Linux and Unix, and third-party versions are available for Windows and other operating systems.

### To write logs to a remote server

1. Set the syslog_destination parameter to the hostname or IP address of the remote server where you want the logs to be written:

```
xe host-param-set uuid=<Xen Cloud Platform_host_uuid> logging:syslog_destination=<hostname
```

2. Issue the command:

```
xe host-syslog-reconfigure uuid=<Xen_Cloud_Platform_host_uuid>
```

to enforce the change. (You can also execute this command remotely by specifying the *host* parameter.)

# Index

## H



## I



## L



## M



## N



## P



## Q



## R

## S



## T



## U



## V



## X