Citrix XenServer ® 6.0 Administrator's Guide

Published Thursday, 15 September 2011
1.0 Edition

Citrix XenServer ® 6.0 Administrator's Guide

Version: 6.0

Citrix, Inc.
851 West Cypress Creek Road
Fort Lauderdale, FL 33309
United States of America

Disclaimers

This document is furnished "AS IS." Citrix, Inc. disclaims all warranties regarding the contents of this document, including, but not limited to, implied warranties of merchantability and fitness for any particular purpose. This document may contain technical or other inaccuracies or typographical errors. Citrix, Inc. reserves the right to revise the information in this document at any time without notice. This document and the software described in this document constitute confidential information of Citrix, Inc. and its licensors, and are furnished under a license from Citrix, Inc.

Trademarks

Contents

Document Overview

This document is a system administrator's guide for Citrix XenServer®, the complete server virtualization platform from Citrix®. It contains procedures to guide you through configuring a XenServer deployment. In particular, it focuses on setting up storage, networking and resource pools, and how to administer XenServer hosts using the xe command line interface.

This document covers the following topics:

- Managing users with Active Directory and Role Based Access Controls
- Creating resource pools and setting up High Availability
- Configuring and managing storage repositories
- Configuring virtual machine memory using Dynamic Memory Control
- Setting control domain memory on a XenServer host
- Configuring networking
- Recovering virtual machines using Disaster Recovery and backing up data
- Monitoring and managing XenServer
- Troubleshooting XenServer
- Using the XenServer xe command line interface

Introducing XenServer

Citrix XenServer® is the complete server virtualization platform from Citrix®. The XenServer package contains all you need to create and manage a deployment of virtual x86 computers running on Xen®, the open-source paravirtualizing hypervisor with near-native performance. XenServer is optimized for both Windows and Linux virtual servers.

XenServer runs directly on server hardware without requiring an underlying operating system, which results in an efficient and scalable system. XenServer works by abstracting elements from the physical machine (such as hard drives, resources and ports) and allocating them to the *virtual machines* running on it.

A virtual machine (VM) is a computer composed entirely of software that can run its own operating system and applications as if it were a physical computer. A VM behaves exactly like a physical computer and contains its own virtual (software-based) CPU, RAM, hard disk and network interface card (NIC).

XenServer lets you create VMs, take VM disk snapshots and manage VM workloads. For a comprehensive list of major XenServer features and editions, visit www.citrix.com/xenserver.

Benefits of Using XenServer

Using XenServer reduces costs by:

- Consolidating multiple VMs onto physical servers
- Reducing the number of separate disk images that need to be managed
- Allowing for easy integration with existing networking and storage infrastructures

Using XenServer increases flexibility by:

- Allowing you to schedule zero downtime maintenance by using XenMotion to live migrate VMs between XenServer hosts
- Increasing availability of VMs by using High Availability to configure policies that restart VMs on another XenServer host if one fails

- Increasing portability of VM images, as one VM image will work on a range of deployment infrastructures

Administering XenServer

There are two methods by which to administer XenServer: XenCenter and the XenServer Command-Line Interface (CLI).

XenCenter is a graphical, Windows-based user interface. XenCenter allows you to manage XenServer hosts, pools and shared storage, and to deploy, manage and monitor VMs from your Windows desktop machine.

The XenCenter *Help* is a great resource for getting started with XenCenter.

The **XenServer Command-line Interface (CLI)** allows you to administer XenServer using the Linux-based xe commands.

For a comprehensive list of xe commands and descriptions, see the *XenServer Administrator's Guide*.

XenServer Editions

The features available in XenServer depend on the edition. The four editions of XenServer are:

- **Citrix XenServer (Free)**: Proven virtualization platform that delivers uncompromised performance, scale, and flexibility at no cost.
- **Citrix XenServer Advanced Edition**: Key high availability and advanced management tools that take virtual infrastructure to the next level.
- **Citrix XenServer Enterprise Edition**: Essential integration and optimization capabilities for production deployments of virtual machines.
- **Citrix XenServer Platinum Edition**: Advanced automation and cloud computing features for enterprise-wide virtual environments.

For more information about how the XenServer edition affects the features available, visit www.citrix.com/xenserver.

New Features in XenServer 6.0

XenServer 6.0 includes a number of new features and ongoing improvements, including:

Integrated Site Recovery (Disaster Recovery):

- Automated remote data replication between storage arrays with fast recovery and failback capabilities. Integrated Site Recovery replaces StorageLink Gateway Site Recovery used in previous versions, removes the Windows VM requirement, and works with any iSCSI or Hardware HBA storage repository.

Integrated StorageLink:

- Access to use existing storage array-based features such as data replication, de-duplication, snapshot and cloning. Replaces the StorageLink Gateway technology used in previous editions and removes the requirement to run a VM with the StorageLink components.

GPU Pass-Through:

- Enables a physical GPU to be assigned to a VM providing high-end graphics. Allows applications to leverage GPU instructions in XenDesktop VDI deployments with HDX 3D Pro.

Virtual Appliance Support (vApp):

- Ability to create multi-VM and boot sequenced virtual appliances (vApps) that integrate with Integrated Site Recovery and High Availability. vApps can be easily imported and exported using the Open Virtualization Format (OVF) standard.

Rolling Pool Upgrade Wizard:

- Simplify upgrades (automated or semi-automated) to XenServer 6.0 with a wizard that performs pre-checks with a step-by-step process that blocks unsupported upgrades.

Microsoft SCVMM and SCOM Support:

- Manage XenServer hosts and VMs with System Center Virtual Machine Manager (SCVMM) 2012. System Center Operations Manager (SCOM) 2012 will also be able to manage and monitor XenServer hosts and VMs. System Center integration is available with a special supplemental pack from Citrix. For more information refer to Microsoft System Center Virtual Machine Manager 2012.

Distributed Virtual Switch Improvements:

- New fail safe mode allows Cross-Server Private Networks, ACLs, QoS, RSPAN and NetFlow settings to continue to be applied to a running VM in the event of vSwitch Controller failure.

Increased Performance and Scale:

- Supported limits have been increased to 1 TB memory for XenServer hosts, and up to16 virtual processors and 128 GB virtual memory for VMs. Improved XenServer Tools with smaller footprint.

Networking Improvements:

- Open vSwitch is now the default networking stack in XenServer 6.0 and now provides formal support for Active-Backup NIC bonding.

VM Import and Export Improvements:

- Full support for VM disk and OVF appliance imports directly from XenCenter with the ability to change VM parameters (virtual processor, virtual memory, virtual interfaces, and target storage repository) with the Import wizard. Full OVF import support for XenServer, XenConvert and VMware.

SR-IOV Improvements:

- Improved scalability and certification with the SR-IOV Test Kit. Experimental SR-IOV with XenMotion support with Solarflare SR-IOV adapters.

Simplified Installer:

- Host installations only require a single ISO.

Enhanced Guest OS Support:

- Support for Ubuntu 10.04 (32/64-bit).
- Updated support for Debian Squeeze 6.0 64-bit, Oracle Enterprise Linux 6.0 (32/64-bit), and SLES 10 SP4 (32/64-bit).
- Experimental VM templates for CentOS 6.0 (32/64-bit) Ubuntu 10.10 (32/64-bit) and Solaris 10.

Workload Balancing Improvements:

- New, ready-to-use Linux-based virtual appliance with a smaller footprint replaces the Windows-based virtual appliance and eliminates the Windows licensing dependency.

XenDesktop Enhancements:

- HDX enhancements for optimized user experience with virtual desktops, GPU Pass-Through, and increased VM and XenServer host limits.

VM Protection and Recovery:

- Now available for Advanced, Enterprise and Platinum Edition customers.

NFS Support for High Availability:

- HA Heartbeat disk can now reside on a NFS storage repository.

XenCenter Improvements:

- XenCenter operations now run in parallel, and XenCenter will be available in Japanese and Simplified Chinese (ETA Q4 2011).

Host Architectural Improvements:

- XenServer 6.0 now runs on the Xen 4.1 hypervisor, provides GPT support and a smaller, more scalable Dom0.

XenServer Documentation

XenServer documentation shipped with this release includes:

- *Release Notes* cover known issues that affect this release.
- *XenServer Quick Start Guide* provides an introduction for new users to the XenServer environment and components. This guide steps through the installation and configuration essentials to get XenServer and the XenCenter management console up and running quickly. After installation, it demonstrates how to create a Windows VM, VM template and pool of XenServer hosts. It introduces basic administrative tasks and advanced features, such as shared storage, VM snapshots and XenMotion live migration.
- *XenServer Installation Guide* steps through the installation, configuration and initial operation of XenServer and the XenCenter management console.
- *XenServer Virtual Machine Installation Guide* describes how to install Windows and Linux VMs within a XenServer environment. This guide explains how to create new VMs from installation media, from VM templates included in the XenServer package and from existing physical machines (P2V). It explains how to import disk images and how to import and export appliances.
- *XenServer Administrator's Guide* gives an in-depth description of the tasks involved in configuring a XenServer deployment, including setting up storage, networking and pools. It describes how to administer XenServer using the xe Command Line Interface.
- *vSwitch Controller User Guide* is a comprehensive user guide to the vSwitch and Controller for XenServer.
- *Supplemental Packs and the DDK* introduces the XenServer Driver Development Kit, which can be used to modify and extend the functionality of XenServer.
- *XenServer Software Development Kit Guide* presents an overview of the XenServer SDK. It includes code samples that demonstrate how to write applications that interface with XenServer hosts.
- *XenAPI Specification* is a reference guide for programmers to the XenServer API.

For additional resources, visit the Citrix Knowledge Center.

Managing Users

Defining users, groups, roles and permissions allows you to control who has access to your XenServer hosts and pools and what actions they can perform.

When you first install XenServer, a user account is added to XenServer automatically. This account is the local super user (LSU), or root, which is authenticated locally by the XenServer computer.

The *local super user (LSU)*, or root, is a special user account used for system administration and has all rights or permissions. In XenServer, the local super user is the default account at installation. The LSU is authenticated by XenServer and not an external authentication service. This means that if the external authentication service fails, the LSU can still log in and manage the system. The LSU can always access the XenServer physical server through SSH.

You can create additional users by adding their Active Directory accounts through either the XenCenter's Users tab or the CLI. All editions of XenServer can add user accounts from Active Directory. However, only XenServer Enterprise and Platinum editions let you assign these Active Directory accounts different levels of permissions (through the Role Based Access Control (RBAC) feature). If you do not use Active Directory in your environment, you are limited to the LSU account.

The permissions assigned to users when you first add their accounts varies according to your version of XenServer:

- In the XenServer and XenServer Advanced edition, when you create (add) new users, XenServer automatically grants the accounts access to all features available in that version.
- In the XenServer Enterprise and Platinum editions, when you create new users, XenServer does not assign newly created user accounts roles automatically. As a result, these accounts do not have any access to the XenServer pool until you assign them a role.

If you do not have one of these editions, you can add users from Active Directory. However, all users will have the Pool Administrator role.

These permissions are granted through roles, as discussed in the section called “Authenticating Users With Active Directory (AD)”.

Authenticating Users With Active Directory (AD)

If you want to have multiple user accounts on a server or a pool, you must use Active Directory user accounts for authentication. This lets XenServer users log in to a pool's XenServers using their Windows domain credentials.

The only way you can configure varying levels of access for specific users is by enabling Active Directory authentication, adding user accounts, and assign roles to those accounts.

Active Directory users can use the xe CLI (passing appropriate `-u` and `-pw` arguments) and also connect to the host using XenCenter. Authentication is done on a per-resource pool basis.

Access is controlled by the use of *subjects*. A subject in XenServer 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 XenCenter or the xe CLI.

If you are familiar with XenCenter, note that the XenServer CLI uses slightly different terminology to refer to Active Directory and user account features:

XenCenter Term	XenServer CLI Term
Users	Subjects
Add users	Add subjects

Understanding Active Directory Authentication in the XenServer Environment

Even though XenServers are Linux-based, XenServer lets you use Active Directory accounts for XenServer user accounts. To do so, it passes Active Directory credentials to the Active Directory domain controller.

When added to XenServer, Active Directory users and groups become XenServer subjects, generally referred to as simply users in XenCenter. When a subject is registered with XenServer, users/groups are authenticated with Active Directory on login and do not need to qualify their user name with a domain name.

> **Note:**
>
> By default, if you did not qualify the user name (for example, enter either mydomain\myuser or myser@mydomain.com), XenCenter always attempts to log users in to Active Directory authentication servers using the domain to which it is currently joined. The exception to this is the LSU account, which XenCenter always authenticates locally (that is, on the XenServer) first.

The external authentication process works as follows:

1. The credentials supplied when connecting to a server are passed to the Active Directory domain controller for authentication.
2. The domain controller checks the credentials. If they are invalid, the authentication fails immediately.
3. If the credentials are valid, the Active Directory controller is queried to get the subject identifier and group membership associated with the credentials.
4. If the subject identifier matches the one stored in the XenServer, the authentication is completed successfully.

When you join a domain, you enable Active Directory authentication for the pool. However, when a pool is joined to a domain, only users in that domain (or a domain with which it has trust relationships) can connect to the pool.

> **Note:**
>
> Manually updating the DNS configuration of a DHCP-configured network PIF is unsupported and might cause Active Directory integration, and consequently user authentication, to fail or stop working.

Upgrading XenServer

When you upgrade from an earlier version of XenServer, any user accounts created in the previous XenServer version are assigned the role of pool-admin. This is done for backwards compatibility reasons. As a result, if you are upgrading from a previous version of XenServer, make sure you revisit the role associated with each user account to make sure it is still appropriate.

Configuring Active Directory Authentication

XenServer supports use of Active Directory servers using Windows 2003 or later.

Active Directory authentication for a XenServer host requires that the same DNS servers are used for both the Active Directory server (configured to allow for interoperability) and the XenServer 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 XenServer host, or by setting values in the PIF objects or using the installer if a manual static configuration is used.

Citrix recommends enabling DHCP to broadcast host names. In particular, the host names `localhost` or `linux` should not be assigned to hosts.

> **Warning:**
>
> XenServer hostnames should be unique throughout the XenServer deployment.

Note the following:

- XenServer labels its AD entry on the AD database using its hostname. Therefore, if two XenServer hosts have the same hostname and are joined to the same AD domain, the second XenServer will overwrite the AD entry of the first XenServer, regardless of if they are in the same or in different pools, causing the AD authentication on the first XenServer to stop working.

  It is possible to use the same hostname in two XenServer hosts, as long as they join different AD domains.
- The XenServer hosts can be in different time-zones, as it is the UTC time that is compared. To ensure synchronization is correct, you may choose to use the same NTP servers for your XenServer pool and the Active Directory server.
- Mixed-authentication pools are not supported (that is, you cannot have a pool where some servers in the pool are configured to use Active Directory and some are not).
- The XenServer Active Directory integration uses the Kerberos protocol to communicate with the Active Directory servers. Consequently, XenServer does not support communicating with Active Directory servers that do not utilize Kerberos.
- For external authentication using Active Directory to be successful, it is important that the clocks on your XenServer hosts are synchronized with those on your Active Directory server. When XenServer joins the Active Directory domain, this will be checked and authentication will fail if there is too much skew between the servers.

> **Warning:**
>
> Host names must consist solely of no more than 63 alphanumeric characters, and must not be purely numeric.

Once you have Active Directory authentication enabled, if you subsequently add a server to that pool, you are prompted to configure Active Directory on the server joining the pool. When you are prompted for credentials on the joining server, enter Active Directory credentials with sufficient privileges to add servers to that domain.

Active Directory integration

Make sure that the following firewall ports are open for outbound traffic in order for XenServer to access the domain controllers.

Port	Protocol	Use
53	UDP/TCP	DNS
88	UDP/TCP	Kerberos 5
123	UDP	NTP
137	UDP	NetBIOS Name Service
139	TCP	NetBIOS Session (SMB)
389	UDP/TCP	LDAP
445	TCP	SMB over TCP
464	UDP/TCP	Machine password changes
3268	TCP	Global Catalog Search

> **Note:**
>
> To view the firewall rules on a Linux computer using *iptables*, run the following command:
> ```
> iptables - nL
> ```

Note:

XenServer uses Likewise (Likewise uses Kerberos) to authenticate the AD user in the AD server, and to encrypt communications with the AD server.

How does XenServer manage the machine account password for AD integration?
Similarly to Windows client machines, Likewise automatically updates the machine account password, renewing it once every 30 days, or as specified in the machine account password renewal policy in the AD server. For more information, refer to http://support.microsoft.com/kb/154501.

Enabling external authentication on a pool

- External authentication using Active Directory can be configured using either XenCenter or 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 used by Active Directory and your XenServer hosts, use you can use these two approaches to setup your DNS:

  1. Set up your domain DNS suffix search order for resolving non-FQDNs:

     ```
     xe pif-param-set uuid=<pif-uuid_in_the_dns_subnetwork> \
         “other-config:domain=suffix1.com suffix2.com suffix3.com”
     ```

  2. Configure the DNS server to use on your XenServer hosts:

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

  3. Manually set the primary 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, Citrix advises that you enable and disable this on a per-pool basis – in this case XenServer 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 XenCenter to disable Active Directory authentication, or the following xe command:

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

User Authentication

To allow a user access to your XenServer 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 XenServer, or a combination of users and groups as your would be appropriate for your authentication requirements. The subject list can be managed from XenCenter or using 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. username 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.

> **Note:**
>
> When using Active Directory groups to grant access for Pool Administrator users who will require host ssh access, the number of users in the Active Directory group must not exceed 500.

Allowing a user access to XenServer using the CLI

- To add an AD subject to XenServer:

  ```
  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 t 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-uuid=<subject-uuid>
   ```
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 XenServer 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 XenCenter or other API sessions that they have already created. In order to terminate these sessions forcefully, XenCenter and the CLI provide facilities to terminate individual sessions, or all currently active sessions. See the XenCenter help for more information on procedures using XenCenter, or 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

> **Warning:**
>
> When you leave the domain (that is, disable Active Directory authentication and disconnect a pool or server from its domain), any users who authenticated to the pool or server with Active Directory credentials are disconnected.

Use XenCenter to leave an AD domain. See the XenCenter help for more information. Alternately 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.

Role Based Access Control

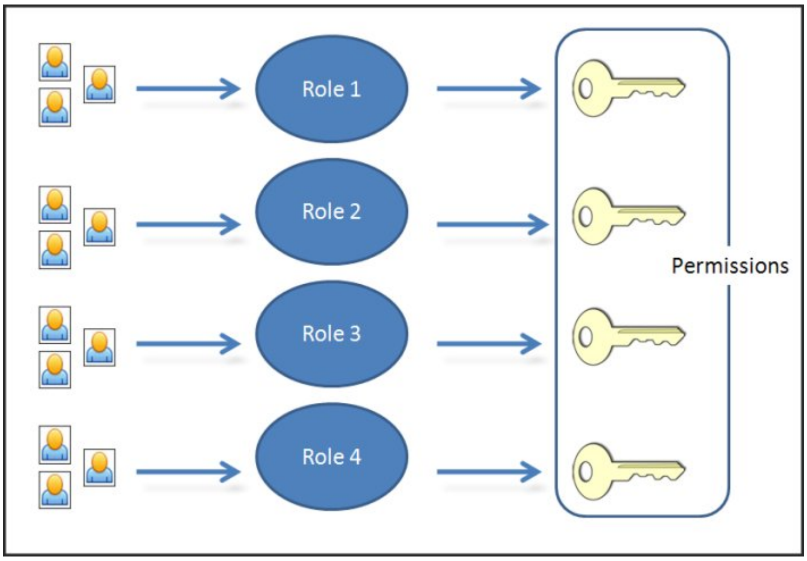
> **Note:**
>
> The full RBAC feature is only available in **Citrix XenServer Enterprise Edition** or higher. To learn more about upgrading XenServer, click here.

XenServer's Role Based Access Control (RBAC) allows you to assign users, roles, and permissions to control who has access to your XenServer and what actions they can perform. The XenServer RBAC system maps a user (or a group of users) to defined roles (a named set of permissions), which in turn have associated XenServer permissions (the ability to perform certain operations).

As users are not assigned permissions directly, but acquire them through their assigned role, management of individual user permissions becomes a matter of simply assigning the user to the appropriate role; this simplifies common operations. XenServer maintains a list of authorized users and their roles.

RBAC allows you to easily restrict which operations different groups of users can perform- thus reducing the probability of an accident by an inexperienced user.

To facilitate compliance and auditing, RBAC also provides an Audit Log feature and its corresponding Workload Balancing Pool Audit Trail report.



RBAC depends on Active Directory for authentication services. Specifically, XenServer keeps a list of authorized users based on Active Directory user and group accounts. As a result, you must join the pool to the domain and add Active Directory accounts before you can assign roles.

The local super user (LSU), or root, is a special user account used for system administration and has all rights or permissions. In XenServer, the local super user is the default account at installation. The LSU is authenticated via XenServer and not external authentication service, so if the external authentication service fails, the LSU can still log in and manage the system. The LSU can always access the XenServer physical host via SSH.

RBAC process

This is the standard process for implementing RBAC and assigning a user or group a role:

1. Join the domain. See Enabling external authentication on a pool
2. Add an Active Directory user or group to the pool. This becomes a subject. See the section called “To Add a Subject to RBAC”.
3. Assign (or modify) the subject's RBAC role. See the section called “To Assign an RBAC Role to a Created subject”.

Roles

XenServer is shipped with the following six, pre-established roles:

- *Pool Administrator* (Pool Admin) – the same as being the local root. Can perform all operations.

  Note:

  The local super user (root) will always have the "Pool Admin" role. The Pool Admin role has the same permissions as the local root.

- *Pool Operator* (Pool Operator) – can do everything apart from adding/removing users and modifying their roles. This role is focused mainly on host and pool management (i.e. creating storage, making pools, managing the hosts etc.)
- *Virtual Machine Power Administrator* (VM Power Admin) – creates and manages Virtual Machines. This role is focused on provisioning VMs for use by a VM operator.
- *Virtual Machine Administrator* (VM Admin) – similar to a VM Power Admin, but cannot migrate VMs or perform snapshots.
- *Virtual Machine Operator* (VM Operator) – similar to VM Admin, but cannot create/destroy VMs – but can perform start/stop lifecycle operations.
- *Read-only* (Read Only) – can view resource pool and performance data.

Note:

You cannot add, remove or modify roles in this version of XenServer.

Warning:

You can not assign the role of *pool-admin* to an AD group which has more than 500 members, if you want users of the AD group to have SSH access.

For a summary of the permissions available for each role and more detailed information on the operations available for each permission, see the section called “Definitions of RBAC Roles and Permissions”.

All XenServer users need to be allocated to an appropriate role. By default, all new users will be allocated to the Pool Administrator role. It is possible for a user to be assigned to multiple roles; in that scenario, the user will have the union of all the permissions of all their assigned roles.

A user's role can be changed in two ways:

1. Modify the subject -> role mapping (this requires the assign/modify role permission, only available to a Pool Administrator.)
2. Modify the user's containing group membership in Active Directory.

Definitions of RBAC Roles and Permissions

The following table summarizes which permissions are available for each role. For details on the operations available for each permission, see Definitions of permissions.

Table 1. Permissions available for each role

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
Assign/ modify roles	X					
Log in to (physical) server consoles (through SSH and XenCenter)	X					
Server backup/ restore	X					
Import/ export OVF/ OVA packages and disk images	X					
Log out active user connections	X	X				
Create and dismiss alerts	X	X				

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
Cancel task of any user	X	X				
Pool management	X	X				
VM advanced operations	X	X	X			
VM create/ destroy operations	X	X	X	X		
VM change CD media	X	X	X	X	X	
View VM consoles	X	X	X	X	X	
XenCenter view mgmt ops	X	X	X	X	X	
Cancel own tasks	X	X	X	X	X	X
Read audit logs	X	X	X	X	X	X
Configure, Initialize, Enable, Disable WLB	X	X				
Apply WLB Optimization Recommendat	X	X				
Modify WLB Report Subscriptions	X	X				
Accept WLB Placement Recommendat	X	X	X			
Display WLB Configuration	X	X	X	X	X	X
Generate WLB Reports	X	X	X	X	X	X

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
Connect to pool and read all pool metadata	X	X	X	X	X	X

Definitions of Permissions

The following table provides additional details about permissions:

Table 2. Definitions of permissions

Permission	Allows Assignee To	Rationale/Comments
Assign/modify roles	• Add/remove users • Add/remove roles from users • Enable and disable Active Directory integration (being joined to the domain)	This permission lets the user grant himself or herself any permission or perform any task. Warning: This role lets the user disable the Active Directory integration and all subjects added from Active Directory.
Log in to server consoles	• Server console access through ssh • Server console access through XenCenter	Warning: With access to a root shell, the assignee could arbitrarily reconfigure the entire system, including RBAC.
Server backup/restore VM create/ destroy operations	• Back up and restore servers • Back up and restore pool metadata	The ability to restore a backup lets the assignee revert RBAC configuration changes.
Import/export OVF/OVA packages and disk images	• Import OVF and OVA packages • Import disk images • Export VMs as OVF/OVA packages	
Log out active user connections	• Ability to disconnect logged in users	
Create/dismiss alerts		Warning: A user with this permission can dismiss alerts for the entire pool. Note: The ability to view alerts is part of the Connect to Pool and read all pool metadata permission.
Cancel task of any user	• Cancel any user's running task	This permission lets the user request XenServer cancel an in-progress task initiated by any user.

Permission	Allows Assignee To	Rationale/Comments
Pool management	• Set pool properties (naming, default SRs) • Enable, disable, and configure HA • Set per-VM HA restart priorities • Enable, disable, and configure Workload Balancing (WLB) • Add and remove server from pool • Emergency transition to master • Emergency master address • Emergency recover slaves • Designate new master • Manage pool and server certificates • Patching • Set server properties • Configure server logging • Enable and disable servers • Shut down, reboot, and power-on servers • System status reports • Apply license • Live migration of all other VMs on a server to another server, due to either WLB, Maintenance Mode, or HA • Configure server management interfaces • Disable server management • Delete crashdumps • Add, edit, and remove networks • Add, edit, and remove PBDs/ PIFs/VLANs/Bonds/SRs • Add, remove, and retrieve secrets	This permission includes all the actions required to maintain a pool. Note: If the primary management interface is not functioning, no logins can authenticate except local root logins.

Permission	Allows Assignee To	Rationale/Comments
VM advanced operations	• Adjust VM memory (through Dynamic Memory Control) • Create a VM snapshot with memory, take VM snapshots, and roll-back VMs • Migrate VMs • Start VMs, including specifying physical server • Resume VMs	This permission provides the assignee with enough privileges to start a VM on a different server if they are not satisfied with the server XenServer selected.
VM create/destroy operations	• Install or delete • Clone VMs • Add, remove, and configure virtual disk/CD devices • Add, remove, and configure virtual network devices • Import/export VMs • VM configuration change	
VM change CD media	• Eject current CD • Insert new CD	
VM change power state	• Start VMs (automatic placement) • Shut down VMs • Reboot VMs • Suspend VMs • Resume VMs (automatic placement)	This permission does not include start_on, resume_on, and migrate, which are part of the VM advanced operations permission.
View VM consoles	• See and interact with VM consoles	This permission does not let the user view server consoles.
Configure, Initialize, Enable, Disable WLB	• Configure WLB • Initialize WLB and change WLB servers • Enable WLB • Disable WLB	When a user's role does not have this permission, this functionality is not visible.
Apply WLB Optimization Recommendations	• Apply any optimization recommendations that appear in the WLB tab	
Modify WLB Report Subscriptions	• Change the WLB report generated or its recipient	

Permission	Allows Assignee To	Rationale/Comments
Accept WLB Placement Recommendations	• Select one of the servers Workload Balancing recommends for placement ("star" recommendations)	
Display WLB Configuration	• View WLB settings for a pool as shown on the WLB tab	
Generate WLB Reports	• View and run WLB reports, including the Pool Audit Trail report	
XenCenter view mgmt operations	• Create and modify global XenCenter folders • Create and modify global XenCenter custom fields • Create and modify global XenCenter searches	Folders, custom fields, and searches are shared between all users accessing the pool
Cancel own tasks	• Lets a user cancel their own tasks	
Read audit log	• Download the XenServer audit log	
Connect to pool and read all pool metadata	• Log in to pool • View pool metadata • View historical performance data • View logged in users • View users and roles • View messages • Register for and receive events	

Note:

In some cases, a Read Only user cannot move a resource into a folder in XenCenter, even after receiving an elevation prompt and supplying the credentials of a more privileged user. In this case, log on to XenCenter as the more privileged user and retry the action.

Using RBAC with the CLI

To List All the Available Defined Roles in XenServer

- Run the command: `xe role-list`

  This command returns a list of the currently defined roles, for example:

```
uuid( RO): 0165f154-ba3e-034e-6b27-5d271af109ba
name ( RO): pool-admin
description ( RO): The Pool Administrator role can do anything

uuid ( RO): b9ce9791-0604-50cd-0649-09b3284c7dfd
name ( RO): pool-operator
description ( RO): The Pool Operator can do anything but access Dom0 \
and manage subjects and roles

uuid( RO): 7955168d-7bec-10ed-105f-c6a7e6e63249
name ( RO): vm-power-admin
description ( RO): The VM Power Administrator role can do anything \
affecting VM properties across the pool

uuid ( RO): aaa00ab5-7340-bfbc-0d1b-7cf342639a6e
name ( RO): vm-admin
description ( RO): The VM Administrator role can do anything to a VM

uuid ( RO): fb8d4ff9-310c-a959-0613-54101535d3d5
name ( RO): vm-operator
description ( RO): The VM Operator role can do anything to an already

uuid ( RO): 7233b8e3-eacb-d7da-2c95-f2e581cdbf4e
name ( RO): read-only
description ( RO): The Read-Only role can only read values
```

Note:

This list of roles is static; it is not possible to add, remove, or modify roles.

To Display a List of Current Subjects:

- Run the command `xe subject-list`

This will return a list of XenServer users, their uuid, and the roles they are associated with:

```
uuid ( RO): bb6dd239-1fa9-a06b-a497-3be28b8dca44
subject-identifier ( RO): S-1-5-21-1539997073-1618981536-2562117463-2244
other-config (MRO): subject-name: example01\user_vm_admin; subject-upn: \
  user_vm_admin@XENDT.NET; subject-uid: 1823475908; subject-gid: 1823474177; \
  subject-sid: S-1-5-21-1539997073-1618981536-2562117463-2244; subject-gecos: \
  user_vm_admin; subject-displayname: user_vm_admin; subject-is-group: false; \
  subject-account-disabled: false; subject-account-expired: false; \
  subject-account-locked: false;subject-password-expired: false
roles (SRO): vm-admin

uuid ( RO): 4fe89a50-6a1a-d9dd-afb9-b554cd00c01a
subject-identifier ( RO): S-1-5-21-1539997073-1618981536-2562117463-2245
other-config (MRO): subject-name: example02\user_vm_op; subject-upn: \
  user_vm_op@XENDT.NET; subject-uid: 1823475909; subject-gid: 1823474177; \
  subject-sid: S-1-5-21-1539997073-1618981536-2562117463-2245; \
  subject-gecos: user_vm_op; subject-displayname: user_vm_op; \
  subject-is-group: false; subject-account-disabled: false; \
  subject-account-expired: false; subject-account-locked: \
  false; subject-password-expired: false
roles (SRO): vm-operator

uuid ( RO): 8a63fbf0-9ef4-4fef-b4a5-b42984c27267
subject-identifier ( RO): S-1-5-21-1539997073-1618981536-2562117463-2242
other-config (MRO): subject-name: example03\user_pool_op; \
  subject-upn: user_pool_op@XENDT.NET; subject-uid: 1823475906; \
  subject-gid: 1823474177; subject-s id:
  S-1-5-21-1539997073-1618981536-2562117463-2242; \
  subject-gecos: user_pool_op; subject-displayname: user_pool_op; \
  subject-is-group: false; subject-account-disabled: false; \
  subject-account-expired: false; subject-account-locked: \
  false; subject-password-expired: false
  roles (SRO): pool-operator
```

To Add a Subject to RBAC

In order to enable existing AD users to use RBAC, you will need to create a subject instance within XenServer, either for the AD user directly, or for one of their containing groups:

1. Run the command `xe subject-add subject-name=<AD user/group>`

This adds a new subject instance.

To Assign an RBAC Role to a Created subject

Once you have added a subject, you can assign it to an RBAC role. You can refer to the role by either its uuid or name:

1. Run the command:

   ```
   xe subject-role-add uuid=<subject uuid> role-uuid=<role_uuid>
   ```

   or

   ```
   xe subject-role-add uuid=<subject uuid> role-name=<role_name>
   ```

   For example, the following command adds a subject with the uuid `b9b3d03b-3d10-79d3-8ed7-a782c5ea13b4` to the Pool Administrator role:

   ```
   xe subject-role-add uuid=b9b3d03b-3d10-79d3-8ed7-a782c5ea13b4 role-name=pool-admin
   ```

To Change a Subject's RBAC Role:

To change a user's role it is necessary to remove them from their existing role, and add them to a new role:

1. Run the commands:

```
xe subject-role-remove uuid=<subject uuid> role-name= \
  <role_name_to_remove>
xe subject-role-add uuid=<subject uuid > role-name= \
  <role_name_to_add>
```

To ensure that the new role takes effect, the user should be logged out and logged back in again (this requires the "Logout Active User Connections" permission - available to a Pool Administrator or Pool Operator).

> **Warning:**
>
> Once you have added or removed a pool-admin subject, there can be a delay of a few seconds for ssh sessions associated to this subject to be accepted by all hosts of the pool.

Auditing

The RBAC audit log will record any operation taken by a logged-in user.

- the message will explicitly record the Subject ID and user name associated with the session that invoked the operation.
- if an operation is invoked for which the subject does not have authorization, this will be logged.
- if the operation succeeded then this is recorded; if the operation failed then the error code is logged.

Audit Log xe CLI Commands

```
xe audit-log-get [since=<timestamp>] filename=<output filename>
```

This command downloads to a file all the available records of the RBAC audit file in the pool. If the optional parameter 'since' is present, then it only downloads the records from that specific point in time.

To Obtain All Audit Records From the Pool

Run the following command:

```
xe audit-log-get filename=/tmp/auditlog-pool-actions.out
```

To Obtain Audit Records of the Pool Since a Precise Millisecond Timestamp

Run the following command:

```
xe audit-log-get since=2009-09-24T17:56:20.530Z \
filename=/tmp/auditlog-pool-actions.out
```

To Obtain Audit Records of the Pool Since a Precise Minute Timestamp

Run the following command:

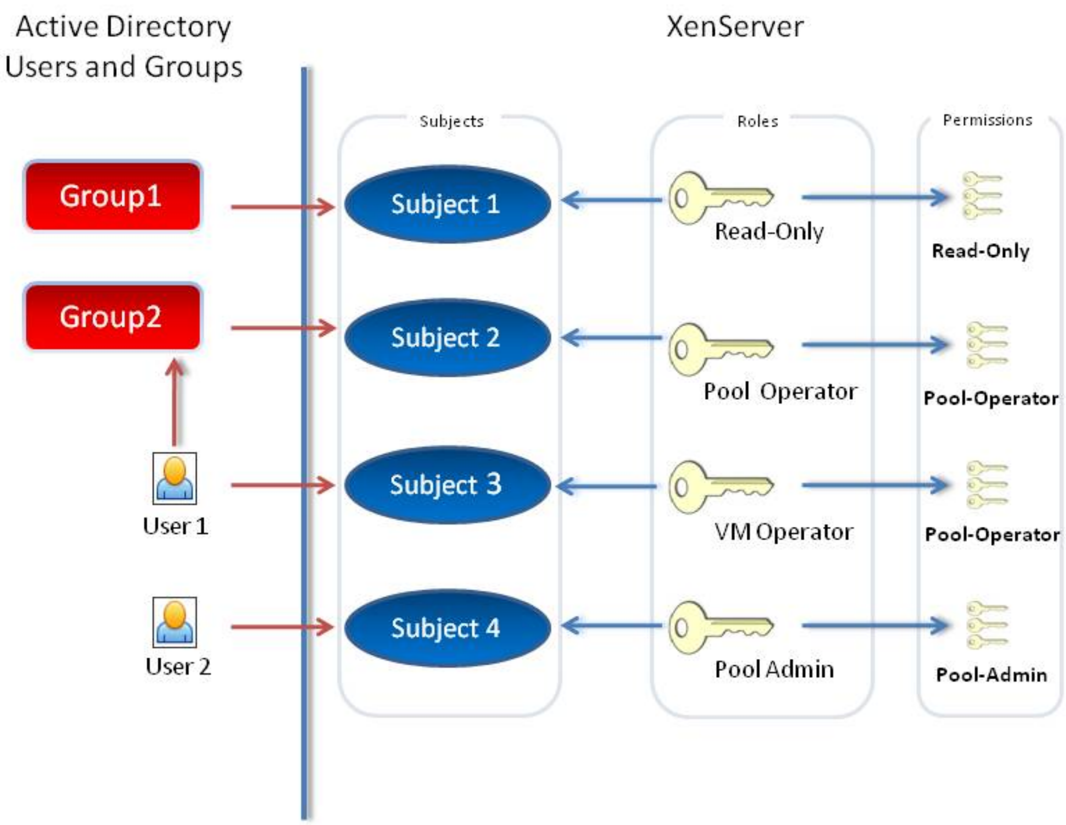
```
xe audit-log-get since=2009-09-24T17:56Z \
filename=/tmp/auditlog-pool-actions.out
```

How Does XenServer Compute the Roles for the Session?

1. The subject is authenticated via the Active Directory server to verify which containing groups the subject may also belong to.

2. XenServer then verifies which roles have been assigned both to the subject, and to its containing groups.

3. As subjects can be members of multiple Active Directory groups, they will inherit all of the permissions of the associated roles.



In this illustration, since Subject 2 (Group 2) is the Pool Operator and User 1 is a member of Group 2, when Subject 3 (User 1) tries to log in, he or she inherits both Subject 3 (VM Operator) and Group 2 (Pool Operator) roles. Since the Pool Operator role is higher, the resulting role for Subject 3 (User 1) is Pool Operator and not VM Operator.

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

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 XenServer Command Line Interface, known as the xe CLI); the master forwards commands to individual members as necessary.

> **Note:**
>
> If the pool's master fails, master re-election will only take place if High Availability is enabled.

Requirements for Creating Resource Pools

A resource pool is a homogeneous (or heterogeneous with restrictions, see the section called “Creating Heterogeneous Resource Pools”) aggregate of one or more XenServer 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 XenServer 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 XenServer 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 primary 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).

XenServer 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 XenServer 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 XenServer host to run a VM and to dynamically move a VM between XenServer hosts) are only available if the pool has one or more shared storage repositories. If possible, postpone creating a pool of XenServer hosts until shared storage is available. Once shared storage has been added, Citrix 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 or XenCenter.

Creating a Resource Pool

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

- 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 primary management interface, which remains the same as the original configuration. For example, if the other pool hosts have their primary management interfaces on a bonded interface, then the joining host must be explicitly migrated to the bond once it has joined.
  - Dedicated storage NICs, which must be re-assigned to the joining host from XenCenter or the CLI, and the PBDs re-plugged to route the traffic accordingly. This is because IP addresses are not assigned as part of the pool join operation, and the storage NIC is not useful without this configured correctly. See the section called “Configuring a dedicated storage NIC” for details on how to dedicate a storage NIC from the CLI.

To join XenServer hosts *host1* and *host2* into a resource pool using the CLI

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

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

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

Naming a resource pool

- XenServer hosts belong to an unnamed pool by default. To create your first resource pool, rename the existing nameless pool. Use tab-complete to find the `pool_uuid`:

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

Creating Heterogeneous Resource Pools

Note:

Heterogeneous resource pool creation is only available to XenServer Advanced editions and above. To learn more about XenServer editions and to find out how to upgrade, visit the Citrix website here.

XenServer 6.0 simplifies expanding deployments over time by allowing disparate host hardware to be joined into a resource pool, known as heterogeneous resource pools. Heterogeneous resource pools are made possible by leveraging technologies in recent Intel (FlexMigration) and AMD (Extended Migration) CPUs that provide CPU "masking" or "leveling". These features allow a CPU to be configured to *appear* as providing a different make, model, or functionality than it actually does. This enables you to create pools of hosts with disparate CPUs but still safely support live migrations.

Using XenServer to mask the CPU features of a new server, so that it will match the features of the existing servers in a pool, requires the following:

- the CPUs of the server joining the pool must be of the same vendor (i.e. AMD, Intel) as the CPUs on servers already in the pool, though the specific type, (family, model and stepping numbers) need not be.
- the CPUs of the server joining the pool must support either Intel FlexMigration or AMD Enhanced Migration.
- the features of the older CPUs must be a sub-set of the features of the CPUs of the server joining the pool.
- the server joining the pool is running the same version of XenServer software, with the same hotfixes installed, as servers already in the pool.
- XenServer Advanced edition or higher.

Creating heterogeneous resource pools is most easily done with XenCenter which will automatically suggest using CPU masking when possible. Refer to the *Pool Requirements* section in the XenCenter help for more details. To display the help in XenCenter press F1.

To add a heterogeneous XenServer host to a resource pool using the xe CLI

1. Find the CPU features of the Pool Master by running the **xe host-get-cpu-features** command.
2. On the new server, run the **xe host-set-cpu-features** command and copy and paste the Pool Master's features into the `features` parameter. For example:

   ```
   xe host-set-cpu-features features=<pool_master's_cpu_ features>
   ```
3. Restart the new server.
4. Run the **xe pool-join** command on the new server to join the pool.

To return a server with masked CPU features back to its normal capabilities, run the **xe host-reset-cpu-features** command.

Note:

To display a list of all properties of the CPUs in a host, run the **xe host-cpu-info** command.

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 XenServer host in the pool.
2. Create the storage repository on *<server:/path>* by issuing the command

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

   The *`device-config:server`* refers to the hostname of the NFS server and *`device-config:serverpath`* refers to the path on the NFS server. Since *`shared`* is set to true, the shared storage will be automatically connected to every XenServer host in the pool and any XenServer hosts that subsequently join will also be connected to the storage. The 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 *Storage* for information about creating other types of shared storage.

Removing a XenServer Host from a Resource Pool

> **Note:**
>
> Before removing a XenServer host from a pool, ensure that you shut down all the VMs running on that host. Otherwise, you may see a warning stating that the host cannot be removed.

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

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 by running the command

   ```
   xe host-list
   ```

3. Eject the required host from the pool:

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

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

   > **Warning:**
   >
   > Do *not* eject a host from a resource pool if it contains important data stored on its local disks. All of the data will be erased upon ejection from the pool. If you wish to preserve this data, copy the VM to shared storage on the pool first using XenCenter, or the **xe vm-copy** CLI command.

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

Preparing a Pool of XenServer Hosts for Maintenance

Before performing maintenance operations on a XenServer host that is part of a resource pool, you should disable it (which prevents any VMs from being started on it), then migrate its VMs to another XenServer host in the pool. This can most readily be accomplished by placing the XenServer host into Maintenance mode using XenCenter. See the XenCenter Help for details.

> **Note:**
>
> Placing the master host into maintenance mode will result in the loss of the last 24hrs of RRD updates for offline VMs. This is because the backup synchronization occurs every 24hrs.

Warning:

Citrix highly recommends rebooting all XenServers prior to installing an update and then verifying their configuration. This is because some configuration changes only take effect when a XenServer is rebooted, so the reboot may uncover configuration problems that would cause the update to fail.

To prepare a XenServer host in a pool for maintenance operations using the CLI

1. Run the command

   ```
   xe host-disable uuid=<xenserver_host_uuid>
   xe host-evacuate uuid=<xenserver_host_uuid>
   ```

   This will disable the XenServer host and then migrate any running VMs to other XenServer hosts in the pool.

2. Perform the desired maintenance operation.

3. Once the maintenance operation is completed, enable the XenServer host:

   ```
   xe host-enable
   ```

   Restart any halted VMs and/or resume any suspended VMs.

High Availability

This section explains the XenServer implementation of virtual machine high availability (HA), and how to configure it using the xe CLI.

Note:

XenServer HA is only available with XenServer Advanced edition or above. To find out about XenServer editions, visit the Citrix website here.

HA Overview

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

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

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

Overcommitting

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

This would happen if there was not enough free memory across the pool to run those VMs following failure. However there are also more subtle changes which can make HA guarantees unsustainable: changes to Virtual

Block Devices (VBDs) and networks can affect which VMs may be restarted on which hosts. Currently it is not possible for XenServer to check all actions before they occur and determine if they will cause violation of HA demands. However an asynchronous notification is sent if HA becomes unsustainable.

Overcommitment Warning

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

Host Fencing

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

Configuration Requirements

> **Note:**
>
> Citrix recommends that you enable HA only in pools that contain at least 3 XenServer hosts. For details on how the HA feature behaves when the heartbeat is lost between two hosts in a pool, see the Citrix Knowledge Base article CTX129721.

To use the HA feature, you need:

- Shared storage, including at least one iSCSI, NFS or Fibre Channel LUN of size 356MB or greater- the *heartbeat SR*. The HA mechanism creates two volumes on the heartbeat SR:

  4MB heartbeat volume
  : Used for heartbeating.

  256MB metadata volume
  : Stores pool master metadata to be used in the case of master failover.

  > **Note:**
  >
  > For maximum reliability, Citrix strongly recommends that you use a dedicated NFS or iSCSI storage repository as your HA heartbeat disk, which must not be used for any other purpose.

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

- A XenServer pool (this feature provides high availability at the server level within a single resource pool).
- XenServer Advanced edition or higher on all hosts.
- Static IP addresses for all hosts.

> **Warning:**
>
> Should the IP address of a server change while HA is enabled, HA will assume that the host's network has failed, and will probably fence the host and leave it in an unbootable state. To remedy this situation, disable HA using the **host-emergency-ha-disable** command, reset the pool master using **pool-emergency-reset-master**, and then re-enable HA.

For a VM to be protected by the HA feature, it must be agile. This means that:

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

Citrix strongly recommends the use of a bonded primary management interface on the servers in the pool if HA is enabled, and multipathed storage for the heartbeat SR.

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

Restart Priorities

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

An explanation of the restart priorities is shown below:

HA Restart Priority	Restart Explanation
0	attempt to start VMs with this priority first
1	attempt to start VMs with this priority, only after having attempted to restart all VMs with priority 0
2	attempt to start VMs with this priority, only after having attempted to restart all VMs with priority 1
3	attempt to start VMs with this priority, only after having attempted to restart all VMs with priority 2
best-effort	attempt to start VMs with this priority, only after having attempted to restart all VMs with priority 3

HA Always Run	Explanation
True	VMs with this setting are included in the restart plan
False	VMs with this setting are NOT included in the restart plan

Warning:

Citrix strongly advises that only StorageLink Service VMs should be given a restart priority of `0`. All other VMs (including those dependent on a StorageLink VM) should be assigned a restart priority `1` or higher.

The "best-effort" HA restart priority must NOT be used in pools with StorageLink SRs.

The restart priorities determine the order in which XenServer attempts to start VMs when a failure occurs. In a given configuration where a number of server failures greater than zero can be tolerated (as indicated in the HA panel in the GUI, or by the `ha-plan-exists-for` field on the pool object on the CLI), the VMs that have restart priorities `0` 1, 2 or 3 are guaranteed to be restarted given the stated number of server failures. VMs with a `best-effort` priority setting are not part of the failover plan and are not guaranteed to be kept running, since capacity is not reserved for them. If the pool experiences server failures and enters a state where the number of tolerable failures drops to zero, the protected VMs will no longer be guaranteed to be restarted. If this condition is reached, a system alert will be generated. In this case, should an additional failure occur, all VMs that have a restart priority set will behave according to the `best-effort` behavior.

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

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

Enabling HA on a XenServer Pool

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

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

Enabling HA Using the CLI

1. Verify that you have a compatible Storage Repository (SR) attached to your pool. iSCSI, NFS or Fibre Channel are compatible SR types. Please refer to the reference guide for details on how to configure such a storage repository using the CLI.
2. For each VM you wish to protect, set a restart priority. You can do this as follows:

   ```
   xe vm-param-set uuid=<vm_uuid> ha-restart-priority=<1> ha-always-run=true
   ```
3. Enable HA on the pool:

   ```
   xe pool-ha-enable heartbeat-sr-uuids=<sr_uuid>
   ```
4. Run the **pool-ha-compute-max-host-failures-to-tolerate** command. This command returns the maximum number of hosts that can fail before there are insufficient resources to run all the protected VMs in the pool.

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

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

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

Removing HA Protection from a VM using the CLI

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

Recovering an Unreachable Host

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

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

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

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

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

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

Shutting Down a host When HA is Enabled

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

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

Shutting Down a VM When it is Protected by HA

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

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

Host Power On

Powering on Hosts Remotely

You can use the XenServer Host Power On feature to turn a server on and off remotely, either from XenCenter or by using the CLI. When using Workload Balancing (WLB), you can configure Workload Balancing to turn hosts on and off automatically as VMs are consolidated or brought back online.

To enable host power, the server must have one of the following power-control solutions:

- **Wake On LAN enabled network card**.
- **Dell Remote Access Cards (DRAC)**. To use XenServer with DRAC, you must install the Dell supplemental pack to get DRAC support. DRAC support requires installing RACADM command-line utility on the server with the remote access controller and enable DRAC and its interface. RACADM is often included in the DRAC management software. For more information, see Dell's DRAC documentation.

- **Hewlett-Packard Integrated Lights-Out (iLO)**. To use XenServer with iLO, you must enable iLO on the host and connect interface to the network. For more information, see HP's iLO documentation.
- A custom script based on the XenAPI that enables you to turn the power on and off through XenServer. For more information, see the section called "Configuring a Custom Script for XenServer's Host Power On Feature".

Using the Host Power On feature requires three tasks:

1. Ensuring the hosts in the pool support controlling the power remotely (that is, they have Wake-on-LAN functionality, a DRAC or iLO card, or you created custom script).
2. Enabling the Host Power On functionality using the CLI or XenCenter.
3. (Optional.) Configuring automatic Host Power On functionality in Workload Balancing. For information on how to configure Host Power On in Workload Balancing please refer to the *Citrix XenServer Workload Balancing Administrator's Guide*.

> **Note:**
>
> You must enable Host Power On and configure the Power Management feature in Workload Balancing before Workload Balancing can turn hosts on and off automatically.

Using the CLI to Manage Host Power On

You can manage the Host Power On feature using either the CLI or XenCenter. This topic provides information about managing it with the CLI.

Host Power On is enabled at the host level (that is, on each XenServer).

After you enable Host Power On, you can turn hosts on using either the CLI or XenCenter.

After enabling Host Power On, you can enable the Workload Balancing Automation and Power Management features, as described in the *Workload Balancing Administrator's Guide*.

To Enable Host Power On Using the CLI

1. Run the command:

   ```
   xe host-set-power-on host=<host uuid>\
   power-on-mode=("" , "wake-on-lan",
   "iLO", "DRAC","custom")
   power-on-config:key=value
   ```

   For iLO and DRAC the keys are *power_on_ip*, *power_on_user*, *power_on_password*. Use *power_on_password* to specify the password if you are using the secret feature.

To Turn on Hosts Remotely Using the CLI

1. Run the command:

   ```
   xe host-power-on host=<host uuid>
   ```

Configuring a Custom Script for XenServer's Host Power On Feature

If your servers' remote-power solution uses a protocol that is not supported by default (such as Wake-On-Ring or Intel Active Management Technology), you can create a custom Linux Python script to turn on your XenServer computers remotely. However, you can also can create custom scripts for iLO, DRAC, and Wake-On-LAN remote-power solutions.

This topic provides information about configuring a custom script for Host Power On using the key/value pairs associated with the XenServer API call *host.power_on*.

When you create a custom script, run it from the command line each time you want to control power remotely on XenServer. Alternatively, you can specify it in XenCenter and use the XenCenter UI features to interact with it.

The XenServer API is documented in the document, the [Citrix XenServer Management API], which is available from the Citrix Web site.

> **Note:**
>
> Do not modify the scripts provided by default in the /etc/xapi.d/plugins/ directory. You can include new scripts in this directory, but you should never modify the scripts contained in that directory after installation.

Key/Value Pairs

To use Host Power On, you must configure the host.power_on_mode and host.power_on_config keys. Their values are provided below.

There is also an API call that lets you set these fields all at once:

void host.set_host_power_on_mode(string mode, Dictionary<string,string> config)

host.power_on_mode

- **Definition**: This contains key/value pairs to specify the type of remote-power solution (for example, Dell DRAC).
- **Possible values**:
  - An empty string, representing power-control disabled
  - "iLO". Lets you specify HP iLO.
  - "DRAC". Lets you specify Dell DRAC. To use DRAC, you must have already installed the Dell supplemental pack.
  - "wake-on-lan". Lets you specify Wake on LAN.
  - Any other name (used to specify a custom power-on script). This option is used to specify a custom script for power management.
- **Type**: string

host.power_on_config

- **Definition**: This contains key/value pairs for mode configuration. Provides additional information for iLO and DRAC.
- **Possible values:**
  - If you configured iLO or DRAC as the type of remote-power solution, you must also specify one of the following keys:
    - "power_on_ip". This is the IP address you specified configured to communicate with the power-control card. Alternatively, you can enter the domain name for the network interface where iLO or DRAC is configured.
    - "power_on_user". This is the iLO or DRAC user name that is associated with the management processor, which you may or may not have changed from its factory default settings.
    - "power_on_password_secret". Specifies using the secrets feature to secure your password.
  - To use the secrets feature to store your password, specify the key "power_on_password_secret".
- **Type**: Map (string,string)

Sample Script

This sample script imports the XenServer API, defines itself as a custom script, and then passes parameters specific to the host you want to control remotely. You must define the parameters *`session`*, *`remote_host`*, and *`power_on_config`* in all custom scripts.

The result is only displayed when the script is unsuccessful.

```
import XenAPI
def custom(session,remote_host,
power_on_config):
result="Power On Not Successful"
for key in power_on_config.keys():
result=result+"
key="+key+"
value="+power_on_config[key]
return result
```

Note:

After creation, save the script in the /etc/xapi.d/plugins with a .py extension.

Storage

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

Storage Overview

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

Storage Repositories (SRs)

XenServer 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 XenServer 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 XenServer 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 XenServer. Similar to SRs, VDIs are persistent, on-disk objects that exist independently of XenServer 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 plug-in interface for each SR, called the SM API.

Physical Block Devices (PBDs)

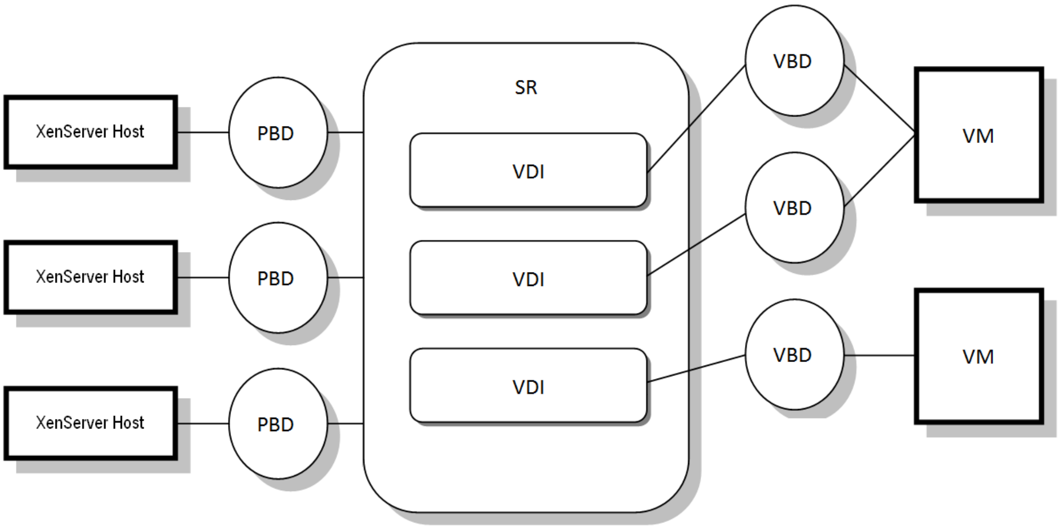
Physical Block Devices represent the interface between a physical server and an attached SR. PBDs are connector objects that allow a given SR to be mapped to a XenServer host. PBDs store the device configuration fields that are used to connect to and interact with a given storage target. For example,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

Virtual 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 plug in (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 XenServer 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 on 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. XenServer supports a maximum chain length of 30, but it is generally not recommended that you approach this limit without good reason. If in doubt, you can always "copy" the VM using XenServer 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, XenServer 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 Utilization

Space utilization 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 utilization 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 utilization differs depending on whether an LVM SR is upgraded or created as a new SR in XenServer. 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 XenServer 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 XenServer 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 Repository Types

The storage repository types supported in XenServer 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 XenServer. 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 **New Storage wizard** in XenCenter. The wizard guides you through the various probing and configuration steps. Alternatively, use 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 XenServer Host in the resource pool.

All XenServer 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, XenServer uses the local disk on the physical host on which it is installed. The Linux Logical Volume Manager (LVM) is used to manage VM storage. A VDI is implemented in VHD format in an LVM logical volume of the specified size.

XenServer versions prior to 6.0 did not use the VHD format and will remain in legacy mode. See the section called "Upgrading LVM Storage from XenServer 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 XenServer CLI.

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

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

ISO

The ISO type handles CD images stored as files in ISO format. This SR type is useful for creating shared ISO libraries. For storage repositories that store a library of ISOs, the `content-type` parameter must be set to `iso`.

For example:

```
xe sr-create host-uuid=<valid_uuid> content-type=iso \
  type=iso name-label=<"Example ISO SR"> \
  device_config:location=<nfs server:path>>
```

Software iSCSI Support

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

XenServer Host iSCSI configuration

All iSCSI initiators and targets must have a unique name to ensure they can be uniquely identified on the network. An initiator has an iSCSI initiator address, and a target has an iSCSI target address. Collectively these are called iSCSI Qualified Names, or IQNs.

XenServer hosts support a single iSCSI initiator which is automatically created and configured with a random IQN during host installation. The single initiator can be used to connect to multiple iSCSI targets concurrently.

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

Note:

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

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

Citrix StorageLink SRs

The Citrix StorageLink (CSL) storage repository provides direct access to native array APIs to offload intensive tasks such as LUN provisioning, snapshot and cloning of data. CSL provides a number of supported adapter types in order to communicate with array management APIs. Once successfully configured, the adapter will handle all provisioning and mapping of storage to XenServer hosts on demand. Data path support for LUNs includes both Fibre Channel and iSCSI as hardware permits.

CSL SRs can be created, viewed, and managed using both XenCenter and the xe CLI.

Note:

For more information on using CSL SR types with XenCenter, see the XenCenter online help.

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

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

CSL supports the following array types:

- NetApp/ IBM N Series

  Important:

  When using NetApp storage with StorageLink on a XenServer host, Initiator Groups are automatically created for the host on the array. These Initiator Groups are created with Linux as the Operating System (OS).

  Manually adding Initiator Groups with other OS values is not recommended.

- Dell EqualLogic PS Series

  Important:

  The Dell EqualLogic API uses SNMP for the communication. It requires SNMP v3 and therefore requires firmware v5.0.0 or greater. If your EqualLogic array uses earlier firmware, you will need to upgrade it to v5.0.0. The firmware can be downloaded from the Dell EqualLogic Firmware download site at https://www.equallogic.com/support/download.aspx?id=1502 (note that you need a Dell Support account to access this page).

  After a firmware upgrade to v5.0.0 or later, you will need to explicitly reset the administrator (grpadmin) password. This is necessary so that the password can be converted to the necessary SNMPv3 authentication and encryption keys.

  To reset the password, log in to the array via telnet or ssh, and run the following command:

  ```
  account select grpadmin passwd
  ```

  At the prompt, enter the new password; at the next prompt, retype it to confirm. The password can be the same as the original password.

Upgrading XenServer with StorageLink SRs

If you are upgrading pools (from XenServer version 5.6 or later to the current version of XenServer) that contain StorageLink Gateway SRs, note that only the following adapters are supported: NetApp and Dell EqualLogic. If the pool contains VMs running on any other types of StorageLink Gateway SRs, do not upgrade the pool.

Note:

Before you upgrade, you will need to first detach any supported StorageLink Gateway SRs and then, once you have upgraded, re-attach them and re-enter your credentials (if you are using XenCenter, the Rolling Pool Upgrade wizard will perform this automatically).

Warning:

If the default SR in the pool you wish to upgrade is a supported StorageLink SR, you must set the default to an SR of a different type (non-StorageLink). Any VMs suspended on a StorageLink Gateway SR by the Rolling Pool Upgrade wizard will not be resumable after the upgrade.

Creating a Shared StorageLink SR

The *`device-config`* parameters for CSL SRs are:

Parameter name	Description	Optional?
target	The server name or IP address of the array management console	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 array management console	Yes
adapterid	The name of the adapter	No
password	The password to use for connecting to the array management console	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

Note:

When creating a new SR on a NetApp array using StorageLink, you can choose to use either an aggregate or an existing FlexVol.

If you choose an existing FlexVol for creating the SR, each VDI would be hosted on a LUN within the FlexVol.

If you choose an aggregate instead, each VDI will be hosted on a LUN inside a new FlexVol within the aggregate.

To Create a CSL SR Using XenCenter

1. On the XenCenter toolbar, click **New Storage. This displays the New Storage Repository wizard.**
2. Under **Virtual disk storage** Select **Advanced StorageLink technology** and then click **Next**.
3. Work through the wizard to configure your specific storage array.

To Create a CSL SR Using the CLI

1. Use the **sr-probe** command with the `device-config:target` parameter and username and password credentials to identify the available storage system IDs.

   For example:

```
xe sr-probe type=cslg device-config:adapterid=NETAPP \
   device-config:username=**** device-config:password=**** \
   device-config:target=****

<csl__storageSystemInfoList>
    <csl__storageSystemInfo>
        <friendlyName>devfiler</friendlyName>
        <displayName>NetApp FAS3020 (devfiler)</displayName>
        <vendor>NetApp</vendor>
        <model>FAS3020</model>
        <serialNum>3064792</serialNum>
        <storageSystemId>NETAPP__LUN__0A50E2F6</storageSystemId>
<systemCapabilities>
    <capabilities>PROVISIONING</capabilities>
    <capabilities>THIN_PROVISIONING</capabilities>
    <capabilities>MAPPING</capabilities>
    <capabilities>MULTIPLE_STORAGE_POOLS</capabilities>
    <capabilities>LUN_GROUPING</capabilities>
    <capabilities>DEDUPLICATION</capabilities>
    <capabilities>DIFF_SNAPSHOT</capabilities>
    <capabilities>REMOTE_REPLICATION</capabilities>
    <capabilities>CLONE</capabilities>
    <capabilities>RESIZE</capabilities>
    <capabilities>REQUIRES_STORAGE_POOL_CLEANUP</capabilities>
    <capabilities>SUPPORTS_OPTIMIZED_ISCSI_LOGIN</capabilities>
    <capabilities>SUPPORTS_INSTANT_CLONE</capabilities>
    <capabilities>SUPPORTS_CLONE_OF_SNAPSHOT</capabilities>
</systemCapabilities>
<protocolSupport>
    <capabilities>FC</capabilities>
    <capabilities>ISCSI</capabilities>
    <capabilities>NFS</capabilities>
    <capabilities>CIFS</capabilities>
</protocolSupport>
<csl__snapshotMethodInfoList>
    <csl__snapshotMethodInfo>
        <name>LUNClone</name>
        <displayName>LUNClone</displayName>
        <maxSnapshots>128</maxSnapshots>
        <supportedNodeTypes><nodeType>STORAGE_VOLUME</nodeType></supportedNodeTypes>
            <snapshotTypeList>
               <snapshotType>DIFF_SNAPSHOT</snapshotType>
               <snapshotType>IS_DEFAULT</snapshotType>
            </snapshotTypeList>
            <snapshotCapabilities>
                <capabilities>THIN_PROVISIONED_TARGET</capabilities>
                <capabilities>AUTO_PROVISIONED_TARGET</capabilities>
            </snapshotCapabilities>
    </csl__snapshotMethodInfo>
    <csl__snapshotMethodInfo>
<name>SplitLUNClone</name>
<displayName>SplitLUNClone</displayName>
<maxSnapshots>128</maxSnapshots>
<supportedNodeTypes><nodeType>STORAGE_VOLUME</nodeType></supportedNodeTypes>
<snapshotTypeList><snapshotType>CLONE</snapshotType></snapshotTypeList>
<snapshotCapabilities><capabilities>THIN_PROVISIONED_TARGET</capabilities>
<capabilities>AUTO_PROVISIONED_TARGET</capabilities>
</snapshotCapabilities>
</csl__snapshotMethodInfo>
</csl__snapshotMethodInfoList>
</csl__storageSystemInfo>
</csl__storageSystemInfoList>
</screen>
```

You can use grep to filter the sr-probe output to just display the storage system IDs:

```
xe sr-probe type=cslg device-config:adapterid=NETAPP \
  device-config:username=xxxx device-config:password=xxxx \
  device-config:target=xxxx | grep storageSystemId

<csl__storageSystemInfoList>
<csl__storageSystemInfo>
<friendlyName>devfiler</friendlyName>
<displayName>NetApp FAS3020 (devfiler)</displayName>
<vendor>NetApp</vendor>
<model>FAS3020</model>
<serialNum>3064792</serialNum>
<storageSystemId>NETAPP__LUN__0A50E2F6</storageSystemId>
<systemCapabilities>
<capabilities>PROVISIONING</capabilities>
```

2. Add the desired storage system ID to the **sr-probe** command to identify the storage pools available within the specified storage system. You can use grep to filter the sr-probe output to just display the storage pool IDs:

```
  xe sr-probe type=cslg device-config:adapterid=NETAPP \
    device-config:username=xxxx device-config:password=xxxx \
    device-config:target=xxxx
    device-config:storageSystemId=NETAPP__LUN__0A50E2F6 | grep storageSystemId

<csl__storagePoolInfo>
    <displayName>aggr0</displayName>
    <friendlyName>aggr0</friendlyName>
    <storagePoolId>61393750-84b6-11dc-9a7d-00a09804ab62</storagePoolId>
    <parentStoragePoolId></parentStoragePoolId>
    <storageSystemId>NETAPP__LUN__0A50E2F6</storageSystemId>
    <sizeInMB>116262</sizeInMB>
    <freeSpaceInMB>5746</freeSpaceInMB>
    <availableFreeSpaceInMB>0</availableFreeSpaceInMB>
    <isDefault>Yes</isDefault>
    <status>0</status>
    <provisioningOptions>
    <supportedRaidTypes><raidType>RAID6</raidType>
```

3. Create the SR specifying the desired storage system and storage pool IDs:

```
xe sr-create type=cslg device-config:adapterid=NETAPP \
  device-config:target=xxxx device-config:username=xxxx \
  device-config:password=xxxx  device-config:storageSystemId=xxxx \
  device-config:storagePoolId=xxxx
```

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 XenServer host, use the following steps to configure the HBA:

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

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

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

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

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. Citrix recommends that only power users perform this process if it is necessary.

Each HBA-based LUN has a corresponding global device path entry under `/dev/disk/by-scsibus` in the format <SCSIid>-<adapter>:<bus>:<target>:<lun> and a standard device path under `/dev`. To remove the device entries for LUNs no longer in use as SRs use the following steps:

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

   ```
   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	Required?
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

Parameter Name	Description	Required?
port	the network port number on which to query the target	no
usediscoverynumber	the specific iSCSI record index to use	no
incoming_chapuser	the username that the iSCSI filter will use to authenticate against the host	no
incoming_chappassword	the password that the iSCSI filter will use to authenticate against the host	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 or XenCenter.

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 XenServer 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 XenServer host to configure the HBA:
   - Emulex: `/bin/sbin/ocmanager`
   - 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-360a9800068666949673446387665336f
        </path>
        <vendor>
            HITACHI
        </vendor>
        <serial>
            730157980002
        </serial>
        <size>
            80530636800
        </size>
        <adapter>
            4
        </adapter>
        <channel>
            0
        </channel>
        <id>
            4
        </id>
        <lun>
            2
        </lun>
        <hba>
            qla2xxx
        </hba>
    </BlockDevice>
    <Adapter>
        <host>
            Host4
        </host>
        <name>
            qla2xxx
        </name>
        <manufacturer>
            QLogic HBA Driver
        </manufacturer>
        <id>
            4
        </id>
    </Adapter>
</Devlist>
```

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

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

Note:

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

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

Creating an NFS SR requires the hostname or IP address of the NFS server. The **sr-probe** command 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 XenServer hosts in the pool, or the creation of the SR and the plugging of the PBD record will fail.

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

XenServer'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.

XenServer 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. XenServer has been tested extensively against Network Appliance FAS270c and FAS3020c storage, using Data OnTap 7.2.2.

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

Warning:

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

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.

XenServer hosts support Fibre Channel (FC) storage area networks (SANs) through Emulex or QLogic host bus adapters (HBAs). All FC configuration required to expose a FC LUN to the host must be completed manually, including storage devices, network devices, and the HBA within the XenServer host. Once all FC configuration is complete the HBA will expose a SCSI device backed by the FC LUN to the host. The SCSI device can then be used to access 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:**
>
> XenServer support for Fibre Channel does not support direct mapping of a LUN to a VM. HBA-based LUNs must be mapped to the host and specified for use in an SR. VDIs within the SR are exposed to VMs as standard block devices.

Storage Configuration

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

Creating Storage Repositories

This section explains how to create Storage Repositories (SRs) of different types and make them available to a XenServer host. The examples provided cover creating SRs using the xe CLI. See the XenCenter help for details on using the **New Storage Repository** wizard to add SRs using XenCenter.

> **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 either XenCenter or the xe CLI.

There are two basic steps involved in creating a new storage repository for use on a XenServer 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 XenServer host and attach it to another. See the section called “Destroying or Forgetting a SR” for details.

> **Note:**
>
> When specifying StorageLink configuration for a XenServer host or pool, supply either the default credentials of username: **`admin`** and password: **`storagelink`**, or any custom credentials specified during installation of the StorageLink Gateway service. Unlike StorageLink Manager, XenCenter does not supply the default credentials automatically.

Upgrading LVM Storage from XenServer 5.0 or Earlier

See the *XenServer 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 (XenServer 5.0 and before) product versions will need to be upgraded before they will support snapshot and fast clone.

> **Warning:**
>
> SR upgrade of SRs created in version 5.0 or before requires the creation of a 4MB metadata volume. Please ensure that there are at least 4MB of free space on your SR before attempting to upgrade the storage.

> **Note:**
>
> Upgrade is a one-way operation so Citrix 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 XenServer 5.5 and later for LVM-based SRs comes with an inherent performance overhead. In cases where optimal performance is desired, XenServer supports creation of VDIs in the *raw* format in addition to the default VHD format. The XenServer 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 XenServer 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 name> sm-config:type=raw
   ```

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 XenServer 5.5 or later) and then copy data into it from an existing volume. Citrix 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>
            149455400000000000000000002000000b70200000f000000
        </SCSIid>
    </LUN>
</iscsi-target>
```

Probing the same target and supplying all three parameters will return a list of SRs that exist on the LUN, if any.

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

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

The following parameters can be probed for each SR type:

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

SR type	device-config parameter, in order of dependency	Can be probed?	Required for sr-create?
	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
	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 XenCenter or on the xe CLI.

> 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 XenServer 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. Therefore, it is unnecessary to use any of the `dm-multipath` CLI tools to query or refresh DM table nodes in XenServer. 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

> **Note:**
>
> Due to incompatibilities with the integrated multipath management architecture, the standard `dm-multipath` CLI utility *should not be used* with XenServer. 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.

MPP RDAC Driver Support for LSI Arrays.

XenServer supports the LSI Multi-Path Proxy Driver (MPP) for the Redundant Disk Array Controller (RDAC). By default this driver is disabled.

To enable the driver:

1. Open a console on the host, and run the following command:

   ```
   # /opt/xensource/libexec/mpp-rdac --enable
   ```

2. Reboot the host.

To disable the driver:

1. Open a console on the host, and run the following command:

   ```
   # /opt/xensource/libexec/mpp-rdac --disable
   ```

2. Reboot the host.

Note:

This procedure must be carried out on every host in a pool.

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 XenServer host, without removing any of the SR contents. In both cases, the PBD of the SR must first be unplugged. Forgetting an SR is the equivalent of the **SR Detach** operation within XenCenter.

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

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

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

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

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

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

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 XenServer 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, and can also be performed using the **Repair Storage Repository** function in XenCenter.

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 XenServer:

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 XenServer 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 and the XenCenter **Repair Storage Repository** feature to convert a local FC SR to a shared FC SR:

1. Upgrade all hosts in the resource pool to XenServer 6.0.
2. Ensure all hosts in the pool have the SR's LUN zoned appropriately. See the section called “Probing an SR” for details on using 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. Within XenCenter the SR is moved from the host level to the pool level, indicating that it is now shared. The SR will be marked with a red exclamation mark to show that it is not currently plugged on all hosts in the pool.
5. Select the SR and then select the **Storage > Repair Storage Repository** menu option.

6. Click **Repair** to create and plug a PBD for each host in the pool.

Moving Virtual Disk Images (VDIs) Between SRs

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

Copying All of a VMs VDIs to a Ddifferent SR

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

1. Shutdown the VM.
2. Within XenCenter select the VM and then select the **VM > Copy VM** menu option.
3. Select the desired target SR.

Copying Individual VDIs to a Different SR

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

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

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

   Note:

   The **vbd-list** command displays both the VBD and VDI UUIDs. Be sure to record the VDI UUIDs rather than the VBD UUIDs.

3. In XenCenter select the VM **Storage** tab. For each VDI to be moved, select the VDI and click the **Detach** button. This step can also be done using the **vbd-destroy** command.

   Note:

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

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

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

5. Within XenCenter select the VM **Storage** tab. Click the **Attach** button and select the VDIs from the new SR. This step can also be done use the `vbd-create` command.
6. To delete the original VDIs, within XenCenter select the **Storage** tab of the original SR. The original VDIs will be listed with an empty value for the VM field and can be deleted with the Delete button.

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.

Automatically Reclaiming Space When Deleting Snapshots

When deleting snapshots with XenServer 6.0, all allocated space allocated on LVM-based SRs is reclaimed automatically and a VM reboot is not required; this is referred to as Online Coalescing.

Note:

Online Coalescing only applies to LVM-based SRs (LVM, LVMoISCSI, and LVMoHBA), it does not apply to EXT or NFS SRs, whose behaviour remains unchanged.

In certain cases, automated space reclamation may be unable to proceed, in these cases it is advisable to use the Off Line Coalesce tool:

- Under conditions where a VM I/O throughput is considerable
- In conditions where space is not being reclaimed after a period of time

Note:

Running the Off Line Coalesce tool will incur some downtime for the VM, due to the suspend/ resume operations performed.

Before running the tool, delete any snapshots and clones you no longer want; the script will reclaim as much space as possible given the remaining snapshots/clones. If you want to reclaim all space, delete all snapshots and clones.

All VM disks must be either on shared or local storage for a single host. VMs with disks in both types of storage cannot be coalesced.

Reclaiming Space Using the Off Line Coalesce Tool

Note:

Online Coalescing only applies to LVM-based SRs (LVM, LVMoISCSI, and LVMoHBA), it does not apply to EXT or NFS SRs, whose behaviour remains unchanged.

Using XenCenter, enable hidden objects (View menu -> Hidden objects). In the Resource pane, select the VM for which you want to obtain the UUID. The UUID will displayed in the General tab.

In the Resource pane, select the resource pool master host (the first host in the list). The UUID will be displayed in the General tab. If you are not using a resource pool, select the VM host.

1. Open a console on the host and run the following command:

   ```
   xe host-call-plugin host-uuid=<host-UUID> \
       plugin=coalesce-leaf fn=leaf-coalesce args:vm_uuid=<VM-UUID>
   ```

   For example, if the VM UUID is 9bad4022-2c2d-dee6-abf5-1b6195b1dad5 and the host UUID is b8722062-de95-4d95-9baa-a5fe343898ea you would run this command:

   ```
   xe host-call-plugin host-uuid=b8722062-de95-4d95-9baa-a5fe343898ea \
       plugin=coalesce-leaf fn=leaf-coalesce args:vm_uuid=9bad4022-2c2d-dee6-abf5-1b6195b1dad5
   ```

2. This command suspends the VM (unless it is already powered down), initiates the space reclamation process, and then resumes the VM.

> **Note:**
>
> Citrix recommends that, before executing the off-line coalesce tool, you shutdown or suspend the VM manually (using either XenCenter or the XenServer CLI). If you execute the coalesce tool on a VM that is running, the tool automatically suspends the VM, performs the required VDI coalesce operation(s), and resumes the VM.
>
> If the Virtual Disk Images (VDIs) to be coalesced are on shared storage, you must execute the off-line coalesce tool on the pool master.
>
> If the VDIs to be coalesced are on local storage, you must execute the off-line coalesce tool on the server to which the local storage is attached.

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>
```

Configuring VM Memory

When a VM is first created, it is allocated a fixed amount of memory. To improve the utilization of physical memory in your XenServer environment, you can use Dynamic Memory Control (DMC), a memory management feature that enables dynamic reallocation of memory between VMs.

XenCenter provides a graphical display of memory usage in its **Memory** tab. This is described in the *XenCenter Help*.

In previous editions of XenServer adjusting virtual memory on VMs required a restart to add or remove memory and an interruption to users' service.

Dynamic Memory Control (DMC) provides the following benefits:

- Memory can be added or removed without restart thus providing a more seamless experience to the user.
- When servers are full, DMC allows you to start more VMs on these servers, reducing the amount of memory allocated to the running VMs proportionally.

What is Dynamic Memory Control (DMC)?

XenServer DMC (sometimes known as "dynamic memory optimization", "memory overcommit" or "memory ballooning") works by automatically adjusting the memory of running VMs, keeping the amount of memory allocated to each VM between specified minimum and maximum memory values, guaranteeing performance and permitting greater density of VMs per server. Without DMC, when a server is full, starting further VMs will fail with "out of memory" errors: to reduce the existing VM memory allocation and make room for more VMs you must edit each VM's memory allocation and then reboot the VM. With DMC enabled, even when the server is full, XenServer will attempt to reclaim memory by automatically reducing the current memory allocation of running VMs within their defined memory ranges.

Without DMC, when a server is full, starting further VMs will fail with "out of memory" errors: to reduce the existing VM memory allocation and make room for more VMs you must edit each VM's memory allocation and then reboot the VM. With DMC enabled, even when the server is full, XenServer will attempt to reclaim memory by automatically reducing the current memory allocation of running VMs within their defined memory ranges.

> **Note:**
>
> Dynamic Memory Control is only available for XenServer Advanced or higher editions. To learn more about XenServer Advanced or higher editions and to find out how to upgrade, visit the Citrix website here.

The Concept of Dynamic Range

For each VM the administrator can set a dynamic memory range – this is the range within which memory can be added/removed from the VM without requiring a reboot. When a VM is running the administrator can adjust the dynamic range. XenServer always guarantees to keep the amount of memory allocated to the VM within the dynamic range; therefore adjusting it while the VM is running may cause XenServer to adjust the amount of memory allocated to the VM. (The most extreme case is where the administrator sets the dynamic min/max to the same value, thus forcing XenServer to ensure that this amount of memory is allocated to the VM.) If new VMs are required to start on "full" servers, running VMs have their memory 'squeezed' to start new ones. The required extra memory is obtained by squeezing the existing running VMs proportionally within their pre-defined dynamic ranges

DMC allows you to configure dynamic minimum and maximum memory levels – creating a Dynamic Memory Range (DMR) that the VM will operate in.

- Dynamic Minimum Memory: A lower memory limit that you assign to the VM.
- Dynamic Higher Limit: An upper memory limit that you assign to the VM.

For example, if the Dynamic Minimum Memory was set at 512 MB and the Dynamic Maximum Memory was set at 1024 MB this would give the VM a Dynamic Memory Range (DMR) of 512 - 1024 MB, within which, it would operate. With DMC, XenServer *guarantees* at all times to assign each VM memory within its specified DMR.

The Concept of Static Range

Many Operating Systems that XenServer supports do not fully 'understand' the notion of dynamically adding or removing memory. As a result, XenServer must declare the maximum amount of memory that a VM will ever be asked to consume at the time that it boots. (This allows the guest operating system to size its page tables and other memory management structures accordingly.) This introduces the concept of a static memory range within XenServer. The static memory range cannot be adjusted while the VM is running. For a particular boot, the dynamic range is constrained such as to be always contained within this static range. Note that the static minimum (the lower bound of the static range) is there to protect the administrator and is set to the lowest amount of memory that the OS can run with on XenServer.

> **Note:**
>
> Citrix advises not to change the static minimum level as this is set at the supported level per operating system – refer to the memory constraints table for more details.
>
> By setting a static maximum level, higher than a dynamic max, means that in the future, if you need to allocate more memory to a VM, you can do so without requiring a reboot.

DMC Behaviour

Automatic VM squeezing

- If DMC is not enabled, when hosts are full, new VM starts fail with 'out of memory' errors.
- If DMC is enabled, even when hosts are full, XenServer will attempt to reclaim memory (by reducing the memory allocation of running VMs within their defined dynamic ranges). In this way running VMs are squeezed proportionally at the same distance between the dynamic minimum and dynamic maximum for all VMs on the host

When DMC is enabled

- When the host's memory is plentiful - All running VMs will receive their Dynamic Maximum Memory level
- When the host's memory is scarce - All running VMs will receive their Dynamic Minimum Memory level.

When you are configuring DMC, remember that allocating only a small amount of memory to a VM can negatively impact it. For example, allocating too little memory:

- Using Dynamic Memory Control to reduce the amount of physical memory available to a VM may cause it to boot slowly. Likewise, if you allocate too little memory to a VM, it may start extremely slowly.
- Setting the dynamic memory minimum for a VM too low may result in poor performance or stability problems when the VM is starting.

How Does DMC Work?

Using DMC, it is possible to operate a guest virtual machine in one of two modes:

1. Target Mode: The administrator specifies a memory target for the guest.XenServer adjusts the guest's memory allocation to meet the target. Specifying a target is particularly useful in virtual server environments, and in any situation where you know exactly how much memory you want a guest to use. XenServer will adjust the guest's memory allocation to meet the target you specify.
2. Dynamic Range Mode: The administrator specifies a dynamic memory range for the guest; XenServer chooses a target from within the range and adjusts the guest's memory allocation to meet the target. Specifying a dynamic range is particularly useful in virtual desktop environments, and in any situation where you want

XenServer to repartition host memory dynamically in response to changing numbers of guests, or changing host memory pressure. XenServer chooses a target from within the range and adjusts the guest's memory allocation to meet the target.

> **Note:**
>
> It is possible to change between target mode and dynamic range mode at any time for any running guest. Simply specify a new target, or a new dynamic range, and XenServer takes care of the rest.

Memory Constraints

XenServer allows administrators to use all memory control operations with any guest operating system. However, XenServer enforces the following memory property ordering constraint for all guests:

```
0 ≤ memory-static-min ≤ memory-dynamic-min ≤ memory-dynamic-max ≤ memory-static-max
```

XenServer allows administrators to change guest memory properties to any values that satisfy this constraint, subject to validation checks. However, in addition to the above constraint, Citrix supports only certain guest memory configurations for each supported operating system. See below for further details.

Supported Operating Systems

Citrix supports only certain guest memory configurations. The range of supported configurations depends on the guest operating system in use. XenServer does not prevent administrators from configuring guests to exceed the supported limit. However, customers are strongly advised to keep memory properties within the supported limits to avoid performance or stability problems.

Operating System			**Supported Memory Limits**		
Family	**Version**	**Architectures**	**Dynamic Minimum**	**Dynamic Maximum**	**Additional Constraints**
Microsoft Windows	XP SP3	x86	≥ 256 MB	≤ 4 GB	Dynamic Minimum ≥ ¼ Static Maximum *for all supported operating systems*
	Server 2003 (+SP1,SP2)	x86	≥ 256 MB	≤ 64 GB	
		x64	≥ 256 MB	≤ 128 GB	
	Server 2008 (+SP2)	x86	≥ 512 MB	≤ 64 GB	
		x64	≥ 512 MB	≤ 128 GB	
	Server 2008 R2 (+SP1)	x64	≥ 512 MB	≤ 128 GB	
	Vista (+SP1,SP2)	x86	≥ 1 GB	≤ 4 GB	
	7 (+SP1)	x86	≥ 1 GB	≤ 4 GB	
		x64	≥ 2 GB	≤ 128 GB	
CentOS Linux	4.5 - 4.8	x86	≥ 256 MB	≤ 16 GB	
	5.0 - 5.6	x86 x64	≥ 512 MB	≤ 16 GB	

Operating System			Supported Memory Limits	
RedHat Enterprise Linux	4.5 - 4.8	x86	≥ 256 MB	≤ 16 GB
	5.0 - 5.6	x86 x64	≥ 512 MB	≤ 16 GB
	6.0	x86	≥512 MB	≤8 GB
		x64	≥512 MB	≤32 GB
Oracle Enterprise Linux	5.0 - 5.6	x86	≥ 512 MB	≤ 64 GB
		x64	≥ 512 MB	≤ 128 GB
	6.0	x86	≥512 MB	≤8 GB
		x64	≥512 MB	≤32 GB
SUSE Enterprise Linux	9 SP4	x86	≥ 256 MB	≤ 16 GB
	10 SP1,SP2,SP3,SP4	x86	≥ 512 MB	≤ 16 GB
		x64	≥ 512 MB	≤ 128 GB
	11 (+SP1)	x86	≥ 512 MB	≤ 16 GB
		x64	≥ 512 MB	≤ 128 GB
Debian GNU/Linux	Lenny (5.0)	x86	≥ 128 MB	≤ 32 GB
	Squeeze (6.0)	x86 x64	≥ 128 MB	≤ 32 GB
Ubuntu	10.04	x86	≥ 128 MB	≤ 512 MB
		x64	≥ 128 MB	≤ 32 GB

Warning:

When configuring guest memory, Citrix advises NOT to exceed the maximum amount of physical memory addressable by your operating system. Setting a memory maximum that is greater than the operating system supported limit, may lead to stability problems within your guest.

In addition reducing the lower limit, below dynamic minimum, could also lead to stability problems. Administrators are encouraged to calibrate the sizes of their VMs carefully, and make sure that their working set of applications functions reliably at dynamic-minimum.

xe CLI Commands

Display the Static Memory Properties of a VM

1. Find the uuid of the required VM:

```
xe vm-list
```

2. Note the uuid, and then run the command **param-name=memory-static**

```
xe vm-param-get uuid=<uuid> param-name=memory-static-{min,max}
```

For example, the following displays the static maximum memory properties for the VM with the uuid beginning ec77:

```
xe vm-param-get uuid= \
  ec77a893-bff2-aa5c-7ef2-9c3acf0f83c0 \
  param-name=memory-static-max;
  268435456
```

This shows that the static maximum memory for this VM is 268435456 bytes (256MB).

Display the Dynamic Memory Properties of a VM

To display the dynamic memory properties, follow the procedure as above but use the command **param-name=memory-dynamic**:

1. Find the uuid of the required VM:

   ```
   xe vm-list
   ```

2. Note the uuid, and then run the command **param-name=memory-dynamic**:

   ```
   xe vm-param-get uuid=<uuid> param-name=memory-dynamic-{min,max}
   ```

   For example, the following displays the dynamic maximum memory properties for the VM with uuid beginning ec77

   ```
   xe vm-param-get uuid= \
     ec77a893-bff2-aa5c-7ef2-9c3acf0f83c0 \
     param-name=memory-dynamic-max;
     134217728
   ```

   This shows that the dynamic maximum memory for this VM is 134217728 bytes (128MB).

Updating Memory Properties

> **Warning:**
>
> It is essential that you use the correct ordering when setting the static/dynamic minimum/maximum parameters. In addition you must not invalidate the following constraint:
>
> `0 ≤ memory-static-min ≤ memory-dynamic-min ≤ memory-dynamic-max ≤ memory-static-max`

Update the static memory range of a virtual machine:

```
xe vm-memory-static-range-set uuid=<uuid> min=<value>max=<value>
```

Update the dynamic memory range of a virtual machine:

```
xe vm-memory-dynamic-range-set \
  uuid=<uuid> min=<value> \
  max=<value>
```

Specifying a target is particularly useful in virtual server environments, and in any situation where you know exactly how much memory you want a guest to use. XenServer will adjust the guest's memory allocation to meet the target you specify. For example:

```
xe vm-target-set target=<value> vm=<vm-name>
```

Update all memory limits (static and dynamic) of a virtual machine:

```
xe vm-memory-limits-set \
  uuid=<uuid> \
  static-min=<value> \
  dynamic-min=<value> \
  dynamic-max=<value> static-max=<value>
```

> **Note:**
>
> - To allocate a specific amount memory to a VM that won't change, set the Dynamic Maximum and Dynamic Minimum to the same value.
> - You cannot increase the dynamic memory of a VM beyond the static maximum.
> - To alter the static maximum of a VM – you will need to suspend or shut down the VM.

Update Individual Memory Properties

> **Warning:**
>
> Citrix advises not to change the static minimum level as this is set at the supported level per operating system – refer to the memory constraints table for more details.

Update the dynamic memory properties of a VM.

1. Find the uuid of the required VM:

   ```
   xe vm-list
   ```

2. Note the uuid, and then use the command **memory-dynamic-{min,max}=<*value*>**

   ```
   xe vm-param-set uuid=<uuid>memory-dynamic-{min,max}=<value>
   ```

The following example changes the dynamic maximum to 128MB:

```
xe vm-param-set uuid=ec77a893-bff2-aa5c-7ef2-9c3acf0f83c0 memory-dynamic-max=128MiB
```

Upgrade Issues

After upgrading from Citrix XenServer 5.5, XenServer sets all VMs memory so that the dynamic minimum is equal to the dynamic maximum.

Workload Balancing Interaction

If Workload Balancing (WLB) is enabled, XenServer defers decisions about host selection to the workload balancing server. If WLB is disabled, or if the WLB server has failed or is unavailable, XenServer will use its internal algorithm to make decisions regarding host selection.

Xen Memory Usage

When calculating the memory footprint of a Xen host there are two components that must be taken into consideration. First there is the memory consumed by the Xen hypervisor itself; then there is the memory consumed by the *control domain* of the host. The control domain is a privileged VM that provides low-level services to other VMs, such as providing access to physical devices. It also runs the management tool stack.

Setting Control Domain Memory

If your control domain requires more allocated memory, this can be set using the Xen CLI.

Use the **xe vm-memory-target-set** command to set the amount of memory available to the control domain.

The **xe vm-memory-target-wait** command can be used to check if the control domain is currently at the requested memory target specified at the last use of the **xe vm-memory-target-set** command. **xe vm-memory-target-wait** will not return until the actual memory usage of the control domain is at the target, or will time out if the target cannot be reached, for example when the target is lower than the actual memory requirements of the VM.

The following fields on a VM define how much memory will be allocated. The default values shown are indicative of a machine with 8 GB of RAM:

name	default	description
memory-actual	411041792	The actual amount of memory current available for use by the VM *Read Only*
memory-target	411041792	The target amount of memory as set by using **xe vm-memory-target-set** *Read Only*
memory-static-max	790102016	The maximum possible physical memory Read Write when the VM is suspended; Read Only when the VM is running
memory-dynamic-max	790102016	The desired maximum memory to be made available Read Write
memory-dynamic-min	306184192	The desired minimum memory to be made available Read Write
memory-static-min	306184192	The minimum possible physical memory Read Write when the VM is suspended; Read Only when the VM is running

name	default	description
memory-overhead	1048576 (for example)	The memory overhead due to virtualization

Dynamic memory values must be within the boundaries set by the static memory values. Additionally the memory target must fall in the range between the dynamic memory values.

> **Note:**
>
> The amount of memory reported in XenCenter on the **General** tab in the **Xen** field may exceed the values set using this mechanism. This is because the amount reported includes the memory used by the control domain, the hypervisor itself, and the crash kernel. The amount of memory used by the hypervisor will be larger for hosts with more memory.

To find out how much host memory is actually available to be assigned to VMs, get the value of the *memory-free* field of the host, and then use the **vm-compute-maximum-memory** command to get the actual amount of free memory that can be allocated to the VM:

```
xe host-list uuid=<host_uuid> params=memory-free
xe vm-compute-maximum-memory vm=<vm_name> total=<host_memory_free_value>
```

Networking

This chapter provides an overview of XenServer networking, including networks, VLANs, and NIC bonds. It also discusses how to manage your networking configuration and troubleshoot it.

> **Important:**
>
> As of this release, the XenServer default network stack is the vSwitch; however, you can revert to the Linux network stack if desired by using the instructions in the section called "vSwitch Networks".

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

- To create networks for standalone XenServer hosts, see the section called "Creating Networks in a Standalone Server".
- To create private networks across XenServer hosts, see the section called "Cross-Server Private networks"
- To create networks for XenServer hosts that are configured in a resource pool, see the section called "Creating Networks in Resource Pools".
- To create VLANs for XenServer hosts, either standalone or part of a resource pool, see the section called "Creating VLANs".
- To create bonds for standalone XenServer hosts, see the section called "Creating NIC Bonds on a Standalone Host".
- To create bonds for XenServer hosts that are configured in a resource pool, see the section called "Creating NIC bonds in resource pools".

For additional information about networking and network design, see *Designing XenServer Network Configurations* in the Citrix Knowledge Center.

For consistency with XenCenter, this chapter now uses the term *primary management interface* to refer to the IP-enabled NIC that carries the management traffic. In previous releases, this chapter used the term *the* management interface. However, *management interface* is now used generically to refer to any IP-enabled NIC, including the NIC carrying management traffic and NICs configured for storage traffic.

Networking Support

XenServer supports up to 16 physical network interfaces (or up to 16 bonded network interfaces) per XenServer host and up to 7 virtual network interfaces per VM.

> **Note:**
>
> XenServer provides automated configuration and management of NICs using the xe command line interface (CLI). Unlike previous XenServer 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.

vSwitch Networks

When used with a controller appliance, vSwitch networks support open flow and provide extra functionality, including Cross Server Private Networks and Access Control Lists (ACL). The controller appliance for the XenServer vSwitch is known as the vSwitch Controller: it lets you monitor your networks through a graphical user interface. The vSwitch Controller:

- Supports fine-grained security policies to control the flow of traffic sent to and from a VM.

- Provides detailed visibility into the behavior and performance of all traffic sent in the virtual network environment.

A vSwitch greatly simplifies IT administration in virtualized networking environments—all VM configuration and statistics remain bound to the VM even if it migrates from one physical host in the resource pool to another. See the *XenServer vSwitch Controller User Guide* for more details.

> **Note:**
>
> To revert to the Linux network stack, run the following command:
>
> ```
> xe-switch-network-backend bridge
> ```
>
> Reboot your host after running this command.
>
> **Warning:**
>
> The Linux network stack is not open-flow enabled, does not support Cross Server Private Networks, and cannot be managed by the XenServer vSwitch Controller.

XenServer Networking Overview

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

One network is created for each physical network interface card (NIC) during XenServer installation. When you add a server to a resource pool, these default networks are merged so that all physical NICs with the same device name are attached to the same network.

Typically, you would only add a new network if you wanted to create an internal network, set up a new VLAN using an existing NIC, or create a NIC bond.

You can configure four different types of networks in XenServer:

- **Single-Server Private networks** have no association to a physical network interface and can be used to provide connectivity between the virtual machines on a given host, with no connection to the outside world.
- **Cross-Server Private networks** extend the single server private network concept to allow VMs on different hosts to communicate with each other by using the vSwitch.
- **External networks** have an association with a physical network interface and provide a bridge between a virtual machine and the physical network interface connected to the network, enabling a virtual machine to connect to resources available through the server's physical network interface card.
- **Bonded networks** create a bond between two NICs to create a single, high-performing channel between the virtual machine and the network.

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

Network Objects

This chapter uses three types of server-side software objects to represent networking entities. These objects are:

- A *PIF*, which represents a physical NIC on a XenServer host. PIF objects have a name and description, a globally unique UUID, the parameters of the NIC that they represent, and the network and server they are connected to.
- A *VIF*, which represents a virtual NIC on a virtual machine. VIF objects have a name and description, a globally unique UUID, and the network and VM they are connected to.

- A *network*, which is a virtual Ethernet switch on a XenServer host. Network objects have a name and description, a globally unique UUID, and the collection of VIFs and PIFs connected to them.

Both 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.

Networks

Each XenServer host has one or more networks, which are virtual Ethernet switches. Networks that are not associated with a PIF are considered *internal* and can be used to provide connectivity only between VMs on a given XenServer host, with no connection to the outside world. Networks associated with a PIF 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 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 primary management interfaces to place management traffic on a desired VLAN. In this case the XenServer host is unaware of any VLAN configuration.

Primary 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 (also known as IP-enabled NICs or simply management interfaces) can be configured to use native VLAN / access mode ports as described above for primary 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

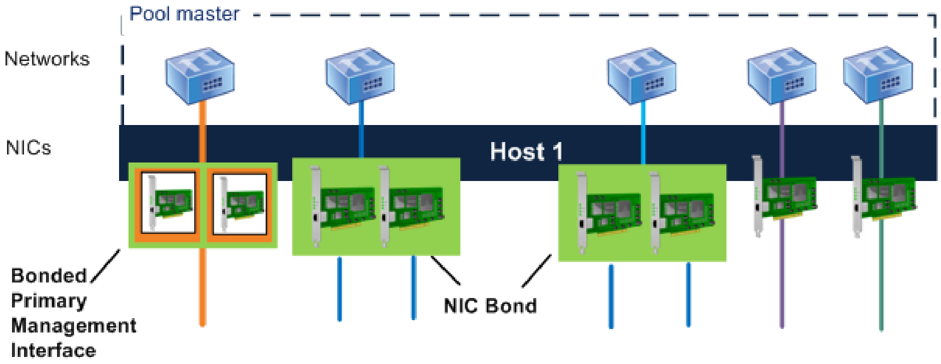
A single switch port can be configured with both trunk and native VLANs, allowing one host NIC to be used for a management interface (on the native VLAN) and for connecting guest VIFs to specific VLAN IDs.

NIC Bonds

NIC bonds can improve XenServer host resiliency by using two physical NICs as if they were one. Specifically, NIC bonding is a technique for increasing resiliency and/or bandwidth in which an administrator configures two NICs together so they logically function as one network card. Both NICs have the same MAC address and, in the case of management interfaces, have one IP address.

If one NIC in the bond fails, the host's network traffic is automatically redirected through the second NIC. NIC bonding is sometimes also known as *NIC teaming*. XenServer supports up to eight bonded networks. You can bond two NICs together of any type (management interfaces or non-management interfaces).

In the illustration that follows, the primary management interface is bonded with a NIC so that it forms a bonded pair of NICs. XenServer will use this bond for management traffic.



This illustration shows three pairs of bonded NICs, including the primary management interface. Excluding the Primary Management Interface bond, XenServer uses the other two NIC bonds and the two un-bonded NICs for VM traffic.

Specifically, you can bond two NICs together of the following types:

- **Primary management interfaces**. You can bond a primary management interface to another NIC so that the second NIC provides failover for management traffic. However, NIC bonding does not provide load balancing for management traffic.
- **NICs (non-management)**. You can bond NICs that XenServer is using solely for VM traffic. Bonding these NICs not only provides resiliency, but doing so also balances the traffic from multiple VMs between the NICs.
- **Other management interfaces**. You can bond NICs that you have configured as management interfaces (for example, for storage). However, for most iSCSI software initiator storage, Citrix recommends configuring multipathing instead of NIC bonding since bonding management interfaces only provides failover and not load balancing.

  It should be noted that certain iSCSI storage arrays, such as Dell EqualLogic, require using bonds.

NIC bonds can work in either an Active/Active mode, with VM traffic balanced between the bonded NICs, or in an Active/Passive mode, where only one NIC actively carries traffic.

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

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

Provided you enable bonding on NICs carrying only guest traffic, both links are active and NIC bonding can balance each VM's traffic between NICs. Likewise, bonding the primary management interface NIC to a second NIC also provides resilience. However, only one link (NIC) in the bond is active and the other remains unused unless traffic fails over to it.

If you bond a management interface, a single IP address is assigned to the bond. That is, each NIC does not have its own IP address; XenServer treats the two NICs as one logical connection.

When bonded NICs are used for VM (guest) traffic, you do not need to configure an IP address for the bond. This is because the bond operates at Layer 2 of the OSI model, the data link layer, and no IP addressing is used at this layer. When used for non-VM traffic (to connect to shared network storage or XenCenter for management), you must configure an IP address for the bond. If you bond a management interface to a non-management NIC, as of XenServer 6.0, the bond assumes the IP address of the management interface automatically.

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

> **Note:**
>
> Bonding is set up with an Up Delay of 31000ms and a Down Delay of 200ms. The seemingly long Up Delay is deliberate because of the time some switches take to actually start routing traffic. Without a delay, when a link comes back after failing, the bond could 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.

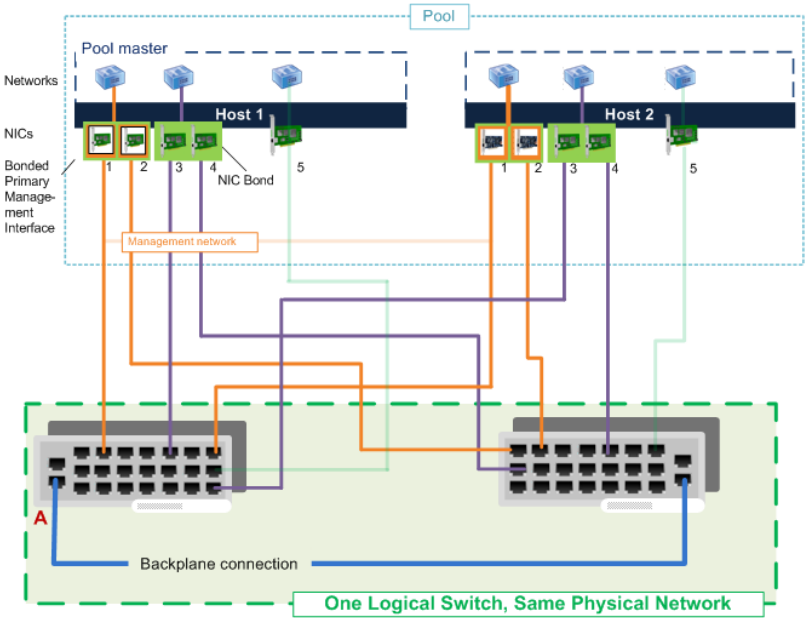
Switch Configuration

Depending on your redundancy requirements, you can connect the NICs in the bond to either the same or separate switches. If you connect one of the NICs to a second, redundant switch and a NIC or switch fails, traffic fails over to the other NIC. Adding a second switch helps in the following ways:

- When you bond NICs used exclusively for VM traffic, traffic is sent over both NICs. If you connect a link to a second switch and the NIC or switch fails, the virtual machines remain on the network since their traffic fails over to the other NIC/switch.
- When you connect one of the links in a bonded primary management interface to a second switch, it prevents a single point of failure for your pool. If the switch fails, the management network still remains online and the hosts can still communicate with each other.

When you attach bonded NICs to two switches, the switches must be running in a stacked configuration. (That is, the switches must be configured to function as a single switch that is seen as a single domain – for example, when multiple rack-mounted switches are connected across the backplane.) Switches must be in a stacked configuration because the MAC addresses of VMs will be changing between switches quite often while traffic is rebalanced across the two NICs. The switches do not require any additional configuration.

The illustration that follows shows how the cables and network configuration for the bonded NICs have to match.



This illustration shows how two NICs in a bonded pair use the same network settings, as represented by the networks in each host. The NICs in the bonds connect to different switches for redundancy.

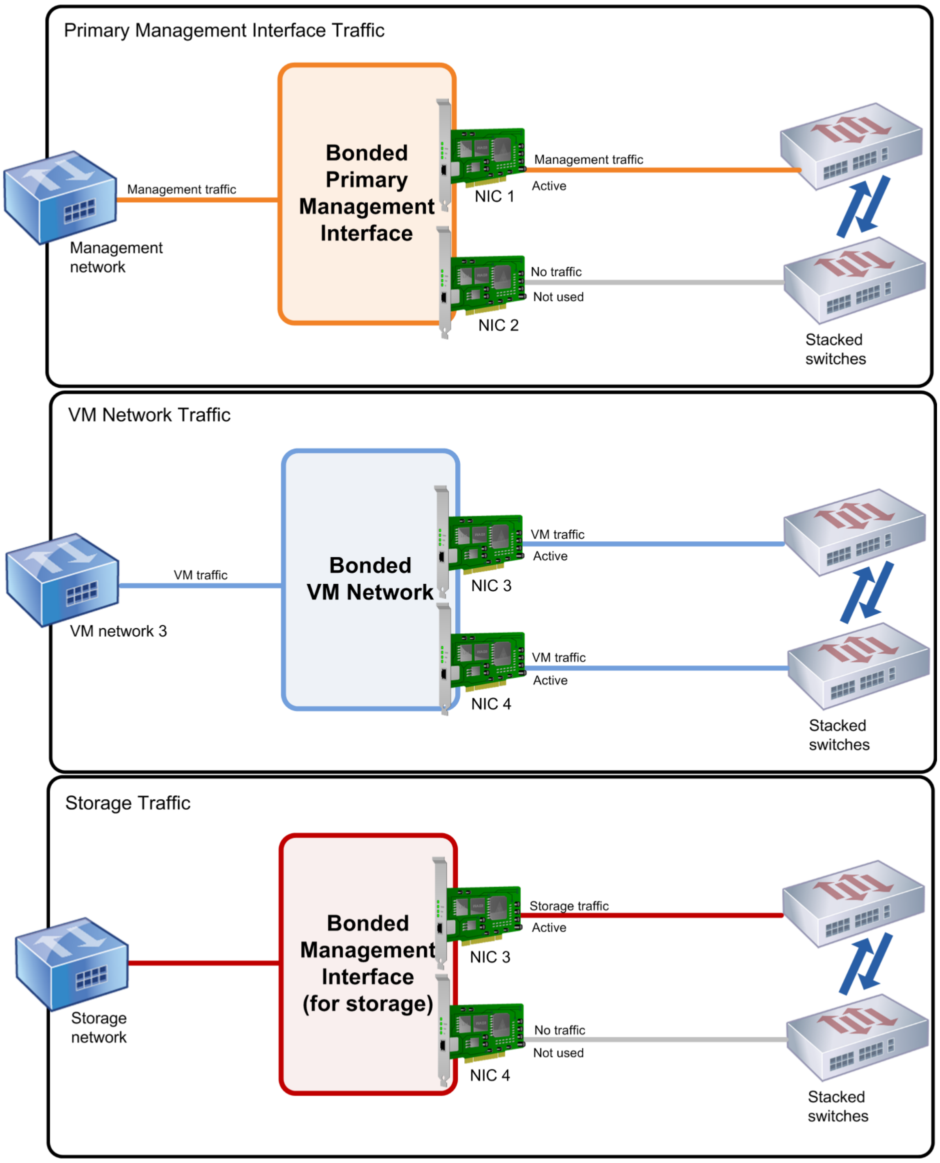
Active-Active Bonding

Active-Active, which is the default bonding mode, is an active/active configuration for guest traffic: both NICs can route VM traffic simultaneously. When bonds are used for management traffic, only one NIC in the bond can route traffic: the other NIC remains unused and provides fail-over support.

For VM traffic, active-active mode also balances traffic. However, it is important to note that "balance" refers to the quantity (MB) of data routed on the NIC. When XenServer rebalances traffic, it simply changes which VM's traffic (more precisely, which VIF's traffic) is carried across which NIC.

While NIC bonding can provide load balancing for traffic from multiple VMs, it cannot provide a single VM with the throughput of two NICs. Any given VIF only uses one of the links in a bond at a time. As XenServer rebalances traffic, VIFs are not permanently assigned to a specific NIC in the bond. However, for VIFs with high throughput, periodic rebalancing ensures that the load on the links is approximately equal.

The illustration that follows shows the differences between the three different types of interfaces that you can bond.



This illustration shows how, when configured in Active-active mode, the links that are active in bonds vary according to traffic type. In the top picture of a management network, NIC 1 is active and NIC 2 is passive. For the VM traffic, both NICs in the bond are active. For the storage traffic, only NIC 3 is active and NIC 4 is passive.

XenServer load balances the traffic between NICs by using the source MAC address of the packet. Because, for management traffic, only one NIC in the bond is used, active-active mode does not balance management traffic.

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

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.

Active-active mode is sometimes referred to as Source Load Balancing (SLB) bonding as XenServer uses SLB to share load across bonded network interfaces. SLB is derived from the open-source ALB mode and reuses the ALB capability to dynamically re-balance load across NICs.

> **Note:**
>
> Active-active bonding does not require switch support for Etherchannel or 802.3ad (LACP).

Active-Passive Bonding

Active-Passive bonding:

- routes traffic over only one of the NICs in the bond
- will failover to use the other NIC in the bond if the active NIC loses network connectivity
- can be configured with one fast link and one slow path for cost savings. In this scenario, the slow path should only be used if there is a failure on the fast path
- does not require switch support for Etherchannel or 802.3ad(LACP)
- is derived from the open source Active-Backup mode

As active-active mode is the default bonding configuration in XenServer, you must configure active-passive mode if you want to use it. You do not need to configure active-passive mode just because a network is carrying management traffic or storage traffic. When bonds are configured or left as active-active mode and XenServer detects management or storage traffic, XenServer automatically leaves one NIC in the bond unused. However, you can explicitly configure active-passive mode, if desired.

When trying to determine when to configure active-passive mode, consider configuring it in situations such as the following:

- When you are connecting one NIC to a switch that does not work well with active-active bonding.

  For example, if the switch does not work well, you might see symptoms like packet loss, an incorrect ARP table on the switch. Likewise, the switch would not update the ARP table correctly and/or the switch would have incorrect settings on the ports (you might configure aggregation for the ports and it would not work).
- When you do not need load balancing or when you only intend to send traffic on one NIC.

  For example, if the redundant path uses a cheaper technology (for example, a lower-performing switch or external up-link) and that results in slower performance, configure active-passive bonding instead.

> **Note:**
>
> As of XenServer 6.0, the vSwitch supports active-passive NIC bonding. If you are using the vSwitch as your networking configuration, you can set the bonding mode to active-passive (also known as active-backup) using the XenCenter or the CLI.

Tip:

Configuring active-passive mode in XenCenter is easy. You simply select **Active-passive** as the bond mode when you create the bond.

Important:

After you have created VIFs or your pool is in production, be extremely careful about making changes to bonds or creating new bonds.

Initial Networking Configuration

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

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 primary 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

When a XenServer 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

In both cases the resulting networking configuration allows connection to the XenServer host by XenCenter, the xe CLI, and any other management software running on separate machines via the IP address of the primary 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 during XenServer installation. 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 sections that follow.

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.

Cross-Server Private networks

Note:

Creating cross-server private networks requires Citrix XenServer Advanced editions or higher. To learn more about XenServer editions, and to find out how to upgrade, visit the Citrix website here.

Previous versions of XenServer allowed you to create single-server private networks that allowed VMs running on the same host to communicate with each other. The *cross-server private network* feature, which extends the single-server private network concept to allow VMs on different hosts to communicate with each other. Cross-server private networks combine the same isolation properties of a single-server private network but with the additional ability to span hosts across a resource pool. This combination enables use of VM agility features such as XenMotion live migration and Workload Balancing (WLB) for VMs with connections to cross-server private networks.

Cross-server private networks are completely isolated. VMs that are not connected to the private network cannot sniff or inject traffic into the network, even when they are located on the same physical host with VIFs connected to a network on the same underlying physical network device (PIF). VLANs provide similar functionality, though unlike VLANs, cross-server private networks provide isolation without requiring configuration of a physical switch fabric, through the use of the Generic Routing Encapsulation (GRE) IP tunnelling protocol.

Private networks provide the following benefits without requiring a physical switch:

- the isolation properties of single-server private networks
- the ability to span a resource pool, enabling VMs connected to a private network to live on multiple hosts within the same pool
- compatibility with features such as XenMotion and Workload Balancing

Cross-Server Private Networks must be created on a management interface, as they require an IP addressable PIF. Any IP-enabled PIF (referred to as a 'Management Interface' in XenCenter) can be used as the underlying network transport. If you choose to put cross-server private network traffic on a second management interface, then this second management interface **must** be on a separate subnet.

If both management interfaces are on the same subnet, traffic will be routed incorrectly.

Note:

To create a cross-server private network, the following conditions must be met:

- All of the hosts in the pool must be using XenServer 6.0 or greater
- All of the hosts in the pool must be using the vSwitch for the networking stack
- The vSwitch Controller must be running and you must have added the pool to it (The pool must have a vSwitch Controller configured that handles the initialization and configuration tasks required for the vSwitch connection)
- The cross-server private network must be created on a NIC configured as a management interface. This can be the primary management interface or another management interface (IP-enabled PIF) you configure specifically for this purpose, provided it is on a separate subnet.

For more information on configuring the vSwitch, see the *XenServer vSwitch Controller User Guide*. For UI-based procedures for configuring private networks, see the XenCenter Help.

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 a private network
- support advanced operations such as VLANs or NIC bonding

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

To add a new network using the CLI

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

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

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

Creating Networks in Resource Pools

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

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

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

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 XenServer host 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

Citrix recommends using XenCenter to create NIC bonds. For instructions, see the XenCenter help.

This section describes how to use the xe CLI to bond NIC interfaces on a XenServer host that is not in a pool. See the section called “Creating NIC bonds in resource pools” for details on using the xe CLI to create NIC bonds on XenServer hosts that comprise a resource pool.

Creating a NIC bond

When you bond a NIC, the bond absorbs the PIF/NIC currently in use as the primary management interface. From XenServer 6.0 onwards, the primary management interface is automatically moved to the bond PIF.

Bonding two NICs

1. 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>
   ```

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

   ```
   xe pif-list
   ```

3. Do one of the following:

   - To configure the bond in active-active mode (default), use the **bond-create** command to create the bond. Using commas to separate the parameters, specify the newly created network UUID and the UUIDs of the PIFs to be bonded:

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

     The UUID for the bond is returned after running the command.

   - To configure the bond in active-passive mode, use the same syntax but add the optional *mode* parameter and specify *active-backup*:

     ```
     xe bond-create network-uuid=<network_uuid> pif-uuids=<pif_uuid_1>,<pif_uuid_2> /
     mode=<balance-slb | active-backup>
     ```

     Note:

     In previous releases, you specified the *other-config:bond-mode* to change the bond mode. While this command still works in XenServer 6.0, it may be not be supported in future releases and it is not as efficient as the *mode* parameter. *other-config:bond-mode* requires running **pif-unplug** and **pif-plug** to get the mode change to take effect.

Controlling the MAC Address of the Bond

When you bond the primary management interface, the PIF/NIC currently in use as the primary management interface is subsumed by the bond. If the host uses DHCP, in most cases the bond's MAC address is the same as the PIF/NIC currently in use, and the primary management interface's IP address can remain unchanged.

You can change the bond's MAC address so that it is different from the MAC address for the (current) primary management-interface NIC. However, as the bond is enabled and the MAC/IP address in use changes, existing network sessions to the host will be dropped.

You can control the MAC address for a bond in two ways:

- An optional *mac* parameter can be specified in the **bond-create** command. You can use this parameter to set the bond MAC address to any arbitrary address.
- If the *mac* parameter is not specified, from XenServer 6.0 onwards, XenServer uses the MAC address of the primary management interface if this is one of the interfaces in the bond. If the primary management interface is not part of the bond, but another management interface is, the bond uses the MAC address (and also the IP address) that management interface. If none of the NICs in the bond are management interfaces, the bond uses the MAC of the first named NIC.

Reverting NIC bonds

If reverting a XenServer host to a non-bonded configuration, be aware that the **bond-destroy** command automatically configures the *primary-slave* as the interface to be used for the primary management interface. Consequently, all VIFs will be moved to the primary management interface.

The term primary-slave refers to the PIF that the MAC and IP configuration was copied from when creating the bond. When bonding two NICs, the primary slave is:

1. The primary management interface NIC (if the primary management interface is one of the bonded NICs).
2. Any other NIC with an IP address (if the primary management interface was not part of the bond).
3. The first named NIC. You can find out which one it is by running the following:

   ```
   xe bond-list params=all
   ```

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 one of the following:

- Using the CLI to configure the bonds on the master and then each member of the pool.
- Using the CLI to configure the bonds on the master and then restarting each member of the pool so that it inherits its settings from the pool master.
- Using XenCenter to configure the bonds on the master. XenCenter automatically synchronizes the networking settings on the member servers with the master, so you do not need to reboot the member servers.

For simplicity and to prevent misconfiguration, Citrix recommends using XenCenter to create NIC bonds. For details, refer to the XenCenter Help.

This section describes using the xe CLI to create bonded NIC interfaces on XenServer hosts that comprise a resource pool. See the section called “Creating a NIC bond” for details on using the xe CLI to create NIC bonds on a standalone XenServer 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 as described in the section called “Creating a NIC bond”.
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. The primary management interface is automatically moved from the host NIC where it was originally configured to the bonded PIF (that is, the primary management interface is now absorbed into the bond so that the entire bond functions as the primary management interface).

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

  ```
  xe host-list
  ```

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:

If you are not using XenCenter for NIC bonding, 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 primary management interface of each host must, however, be manually reconfigured.

Follow the procedure in previous sections to create a NIC Bond, see the section called “Adding NIC bonds to new resource pools”.

Configuring a dedicated storage NIC

You can use either XenCenter or the xe CLI to assign a NIC an IP address and dedicate it to a specific function, such as storage traffic. When you configure a NIC with an IP address, you do so by creating a management interface. IP-enabled NICs are referred to collectively as *management interfaces*. (The IP-enabled NIC XenServer used for management is a *type* of management interface known as the *primary* management interface.)

When you want to dedicate a management interface for a specific purpose, you must ensure the appropriate network configuration is in place to ensure the NIC is used only for the desired traffic. For example, to dedicate a NIC to storage traffic, the NIC, storage target, switch, and/or VLAN must be configured so that the target is only accessible over the assigned NIC. If your physical and IP configuration do not limit the traffic that can be sent across the storage management interface, it is possible for other traffic, such as management traffic, to be sent across the storage NIC.

Note:

When selecting a NIC to configure as a management interface for use with iSCSI or NFS SRs, ensure that the dedicated NIC uses a separate IP subnet that is not routable from the primary management interface. If this is not enforced, then storage traffic may be directed over the main primary 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 primary management interface also (bearing in mind that this configuration is not the best practice), 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 restarts the storage connection and routes it over the correct interface.
- Alternatively, you can use **xe pif-forget** to remove the interface from the XenServer database and manually configure it in the control domain. This is an advanced option and requires you to be familiar with how to manually configure Linux networking.

Using SR-IOV Enabled NICs

Single Root I/O Virtualization (SR-IOV) is a PCI device virtualization technology that allows a single PCI device to appear as multiple PCI devices on the physical PCI bus. The actual physical device is known as a **Physical Function** (PF) while the others are known as **Virtual Functions** (VF). The purpose of this is for the hypervisor to directly assign one or more of these VFs to a Virtual Machine (VM) using SR-IOV technology: the guest can then use the VF as any other directly assigned PCI device.

Assigning one or more VFs to a VM allows the VM to directly exploit the hardware. When configured, each VM behaves as though it is using the NIC directly, reducing processing overhead and improving performance.

> **Warning:**
>
> If your VM has an SR-IOV VF, functions that require VM mobility, for example, Live Migration, Workload Balancing, Rolling Pool Upgrade, High Availability and Disaster Recovery, are not possible. This is because the VM is directly tied to the physical SR-IOV enabled NIC VF. In addition, VM network traffic sent via an SR-IOV VF bypasses the vSwitch, so it is not possible to create ACLs or view QoS.

Assigning a SR-IOV NIC VF to a VM

> **Note:**
>
> SR-IOV is supported only with SR-IOV enabled NICs listed on the XenServer Hardware Compatibility List and only when used in conjunction with a Windows Server 2008 guest operating system.

1. Open a local command shell on your XenServer host.
2. Run the command `lspci` to display a list of the Virtual Functions (VF). For example:

   ```
   07:10.0 Ethernet controller: Intel Corporation 82559 \
     Ethernet Controller Virtual Function (rev 01)
   ```

   In the example above, `07:10.0` is the **bus:device.function** address of the VF.
3. Assign the required VF to the target VM by running the following commands:

   ```
   xe vm-param-set other-config:pci=0/0000:<bus:device.function> uuid=<vm-uuid>
   ```
4. Start the VM, and install the appropriate VF driver for your specific hardware.

   > **Note:**
   >
   > You can assign multiple VFs to a single VM, however the same VF cannot be shared across multiple VMs.

Controlling the rate of outgoing data (QoS)

To limit the amount of *outgoing* data a VM can send per second, you can set an optional Quality of Service (QoS) value on VM virtual interfaces (VIFs). The setting lets you specify a maximum transmit rate for outgoing packets in kilo*bytes* per second.

The QoS value limits the rate of transmission *from* the VM. The QoS setting does not limit the amount of data the VM can receive. If such a limit is desired, Citrix recommends limiting the rate of incoming packets higher up in the network (for example, at the switch level).

Depending on networking stack configured in the pool, you can set the Quality of Service (QoS) value on VM virtual interfaces (VIFs) in one of two places—either a) on the vSwitch Controller or b) in XenServer (using the CLI or XenCenter)—as described in the following table:

Networking Stack	Configuration Methods Available
vSwitch	• **vSwitch Controller**. This is the preferred method of setting the maximum transmission rate on a VIF when the vSwitch is the networking stack. When using the vSwitch stack, the XenCenter QoS option is not available. • **xe commands**. It is possible to set the QoS transmit rate using the commands in the example that follows. However, the preferred method is through the vSwitch Controller UI, which provides more finely grained control.
Linux bridge	• **XenCenter**. You can set the QoS transmit rate limit value in the properties dialog for the virtual interface. • **xe commands**. You can set the QoS transmit rate using the CLI using the commands in the section that follow.

Important:

When the vSwitch is configured as the networking stack, it is possible to inadvertently configure a QoS value on the vSwitch Controller *and* inside of the XenServer host. In this case, XenServer limits the outgoing traffic using the lowest rate that you set.

Example of CLI command for QoS:

To limit a VIF to a maximum transmit rate of 100 kilobytes per second using the CLI, 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
```

Note:

If you are using the vSwitch Controller, Citrix recommends setting the transmission rate limit in the vSwitch Controller instead of this CLI command. For directions on setting the QoS rate limit in the vSwitch Controller, see the *vSwitch Controller User Guide*.

Changing networking configuration options

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

- changing the hostname (that is, the Domain Name System (DNS) name)
- adding or removing DNS servers
- changing IP addresses
- changing which NIC is used as the primary management interface
- adding a new physical NIC to the server

Hostname

The system hostname, also known as the domain or DNS name, 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 host-uuid=<host_uuid> host-name=<host-name>
```

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 XenServer host, use the **pif-reconfigure-ip** command. For example, for a PIF with a static IP:

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

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

XenServer hosts in resource pools have a single management IP address used for management and communication to and from other hosts in the pool. The steps required to change the IP address of a host's primary 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 **Repair Storage** function in XenCenter, or 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 member host (not pool master)

1. Use the **pif-reconfigure-ip** CLI command to set the IP address as desired. See Appendix A, *Command Line Interface* for details on the parameters of the **pif-reconfigure-ip** command:

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

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

   ```
   xe host-list
   ```

Changing the IP address of the master XenServer host requires additional steps because each of the member hosts uses the 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 the pool master

1. Use the **pif-reconfigure-ip** CLI command to set the IP address as desired. See Appendix A, *Command Line Interface* for details on the parameters of the **pif-reconfigure-ip** command:

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

2. When the IP address of the pool master host is changed, all member hosts will enter into an emergency mode when they fail to contact the master host.
3. On the master XenServer host, use the **pool-recover-slaves** command to force the master to contact each of the member hosts and inform them of the new master IP address:

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

Primary management interface

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

To change the NIC used for the primary management interface

1. Use the **pif-list** command to determine which PIF corresponds to the NIC to be used as the primary 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 primary management interface. If necessary, use the **pif-reconfigure-ip** command to configure IP addressing for the PIF to be used. See Appendix A, *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 primary 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 primary 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.

> **Warning:**
>
> Once the primary management interface is disabled, you will have to log in on the physical host console to perform management tasks and external interfaces such as XenCenter will no longer work.

Adding a new physical NIC

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

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 XenServer host.

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

- Citrix recommends that you ensure networking configuration is set up correctly before creating a resource pool, as it is usually easier to recover from a bad configuration in a non-pooled state.
- You can run the **host-management-reconfigure** command from any host in the pool. This is because you specify the PIF UUID for the NIC that you want to affect.
- The **host-management-disable** command affects the XenServer host on which it is run so it is not suitable for use on one host in a pool to change the configuration of another. Run this command directly on the console of the XenServer host to be affected, or use the **xe** *-s*, *-u*, and *-pw* remote connection options.
- When the `xapi` service starts, it applies configurations to the primary management interface first. The name of the primary 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 primary 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).

Disaster Recovery and Backup

The XenServer Disaster Recovery (DR) feature is designed to allow you to recover virtual machines (VMs) and vApps from a catastrophic failure of hardware which disables or destroys a whole pool or site. For protection against single server failures, see the section called “High Availability”

> **Note:**
>
> Disaster Recovery is only available for XenServer Platinum edition. To learn more about XenServer Advanced or higher editions and to find out how to upgrade, visit the Citrix website here.
>
> You must be logged in as *root* or have the role of *Pool Operator* or higher to use this feature.

Understanding XenServer DR

XenServer DR works by storing all the information needed to recover your business-critical VMs and vApps on storage repositories (SRs) that are then replicated from your primary (production) environment to a backup environment. When a protected pool at your primary site goes down, the VMs and vApps in that pool can be recovered from the replicated storage and recreated on a secondary (DR) site, with minimal application or user downtime.

> **Note:**
>
> Citrix strongly recommends using the new XenServer 6.0 Disaster Recovery feature, as the legacy Metadata Backup, Restore and Update mechanism (accessible via the XenServer host console) will be depreciated in a future XenServer release. Citrix advises customers using the legacy mechanism to migrate to the new, integrated feature.

In the event of a disaster, the **Disaster Recovery** wizard in XenCenter can be used to interrogate this storage and import chosen VMs and vApps into a recovery pool. Once the VMs are running in the recovery pool, the recovery pool metadata is also replicated to allow any changes to VM settings to be populated back to the primary pool, should the primary pool be recovered. If the XenCenter wizard finds information for the same VM present in two or more places (for example, storage from the primary site, storage from the disaster recovery site and also in the pool that the data is to be imported into) then the wizard will ensure that only the most recent information per Virtual Machine is used.

The Disaster Recovery feature can be used both with XenCenter and the xe CLI. See the section called “Disaster Recovery (DR) Commands” for details on these commands.

> **Tip:**
>
> You can also use the Disaster Recovery wizard to run test failovers for non-disruptive testing of your disaster recovery system. In a test failover, all the steps are the same as for failover, but the VMs and vApps are not started up after they have been recovered to the DR site, and cleanup is performed when the test is finished to remove all VMs, vApps and storage recreated on the DR site.

XenServer VMs consist of two components:

- Virtual disks that are being used by the VM, stored on configured storage repositories (SRs) in the pool where the VMs are located.
- Metadata describing the VM environment. This is all the information needed to recreate the VM if the original VM is unavailable or corrupted. Most metadata configuration data is written when the VM is created and is updated only when you make changes to the VM configuration. For VMs in a pool, a copy of this metadata is stored on every server in the pool.

In a DR environment, VMs are recreated on a secondary (DR) site from the pool metadata – configuration information about all the VMs and vApps in the pool. The metadata for each VM includes its name, description and Universal Unique Identifier (UUID), and its memory, virtual CPU and networking and storage configuration. It also includes the VM startup options – start order, delay interval and HA restart priority – which are used when restarting the VM in an HA or DR environment. For example, when recovering VMs during disaster recovery, the VMs within a vApp will be restarted in the DR pool in the order specified in the VM metadata, and with the specified delay intervals.

DR Infrastructure Requirements

To use XenServer DR, the appropriate DR infrastructure needs to be set up at both the primary and secondary sites:

- The storage used for both the pool metadata *and* the virtual disks used by the VMs must be replicated from your primary (production) environment to a backup environment. Storage replication, for example using mirroring, is best handled by your storage solution, and will vary from device to device.
- Once VMs and vApps have been recovered to a pool on your DR site and are up and running, the SRs containing the DR pool metadata and virtual disks must also be replicated to allow the recovered VMs and vApps to be restored back to the primary site *(failed back)* once the primary site is back online.
- The hardware infrastructure at your DR site does not have to match the primary site, but the XenServer environment must be at the same release and patch level, and sufficient resources should be configured in the target pool to allow all the failed over VMs to be recreated and started.

Warning:

The Disaster Recovery wizard does **not** control any Storage Array functionality.

Users of the Disaster Recovery feature **must** ensure that the metadata storage is, in some way replicated between the two sites. Some Storage Arrays contain "Mirroring" features to achieve the replication automatically, if these features are used then it is essential that the mirror functionality is disabled ("mirror is broken") before Virtual Machines are restarted on the recovery site.

Deployment Considerations

Please review the following steps before enabling Disaster Recovery.

Steps to Take Before a Disaster

The following section describes the steps to take before disaster.

- Configure your VMs and vApps.
- Note how your VMs and vApps are mapped to SRs, and the SRs to LUNs. Take particular care with the naming of the `name_label` and `name_description` fields. Recovering VMs and vApps from replicated storage will be easier if your SRs are named in a way that captures how your VMs and vApps are mapped to SRs, and the SRs to LUNs.
- Arrange replication of the LUNs.
- Enable pool metadata replication to one or more SRs on these LUNs.

Steps to Take After a Disaster

The following section describes the steps to take after a disaster has occurred.

- Break any existing storage mirrors so that the recovery site has read/write access to the shared storage.
- Ensure that the LUNs you wish to recover VM data from are not attached to any other pool, or corruption may occur.
- If you want to protect the *recovery* site from a disaster, you must enable pool metadata replication to one or more SRs on the recovery site.

Steps to Take After a Recovery

The following section describes the steps to take after a successful recovery of data.

- Re-synchronise any storage mirrors.
- On the recovery site, shutdown down cleanly the VMs or vApps that you want to move back to the primary site.
- On the primary site, follow the same procedure as for the failover above, to failback selected VMs or vApps to the primary
- To protect the primary site against future disaster - you must re-enable pool metadata replication to one or more SRs on the replicated LUNs.

Enabling Disaster Recovery in XenCenter

This section describes how to enable Disaster Recovery in XenCenter. Use the **Configure DR** dialog box to identify storage repositories (SRs) where the metadata for a pool – configuration information about all the VMs and vApps in the pool – will be stored. This metadata will be updated whenever you make changes to VM or vApp configuration within the pool.

> **Note:**
>
> Disaster Recovery can only be enabled when using LVM over HBA or LVM over iSCSI. A small amount of space will be required on this storage for a new LUN which will contain the pool recovery information.

To do this:

1. On the primary site, select the pool that you want to protect. From the **Pool** menu, point to **Disaster Recovery**, and then click **Configure**.
2. Select up to 8 SRs where the pool metadata will be stored. A small amount of space will be required on this storage for a new LUN which will contain the pool recovery information.

   > **Note:**

Information for all VMs in the pool is stored, VMs do not need to be independently selected for protection.

3. Click **OK**. Your pool is now protected.

Recovering VMs and vApps in the Event of Disaster (Failover)

This section explains how to recover your VMs and vApps on the secondary (recovery) site.

1. In XenCenter select the secondary pool, and on the **Pool** menu, click **Disaster Recovery** to open the **Disaster Recovery** wizard.

   This wizard gives three recovery options: **Failover**, **Failback**, and **Test Failover**. To recover on to your secondary site, select **Failover** and then click **Next**.

   > **Warning:**
   >
   > If you use Fibre Channel shared storage with LUN mirroring to replicate the data to the secondary site, before you attempt to recover data, **mirroring must be broken** so that the secondary site has Read/Write access.

2. Select the storage repositories (SRs) containing the pool metadata for the VMs and vApps that you want to recover.

   By default, the list on this wizard page shows all SRs that are currently attached within the pool. To scan for more SRs, choose **Find Storage Repositories** and then select the storage type to scan for:

   - To scan for all the available Hardware HBA SRs, select **Find Hardware HBA SRs**.
   - To scan for software iSCSI SRs, select **Find Software iSCSI SRs** and then enter the target host, IQN and LUN details in the dialog box.

   When you have selected the required SRs in the wizard, click **Next** to continue.

3. Select the VMs and vApps that you wish to recover and choose the appropriate **Power state after recovery** option to specify whether you want the wizard to start them up automatically as soon as they have been recovered, or whether you prefer to wait and start them up manually yourself after failover is complete.

   Click **Next** to progress to the next wizard page and begin failover prechecks.

4. The wizard performs a number of prechecks before starting failover, for example, to ensure that all the storage required by the selected VMs and vApps is available. If any storage is missing at this point, you can click **Attach SR** on this page to find and attach the relevant SR.

   Resolve any issues on the prechecks page, and then click **Failover** to begin the recovery process.

5. A progress page is displayed showing whether recovery was successful for each VM and vApp. Failover may take some time depending on the number of VMs and vApps you are recovering, as the metadata for the VMs and vApps are exported from the replicated storage. The VMs and vApps are recreated in the primary pool, the SRs containing the virtual disks are attached to the recreated VMs, and VMs are started, if specified.

6. When the failover is complete, click **Next** to see the summary report. Click **Finish** on the summary report page to close the wizard.

Once the primary site is available again, and you want to return to running your VMs on that site, work through the Disaster Recovery wizard again, but this time select the **Failback** option.

Restoring VMs and vApps to the Primary Site After Disaster (Failback)

This section explains how to restore VMs and vApps from replicated storage back to a pool on your primary (production) site when the primary site comes back up after a disaster event. To failback VMs and vApps to your primary site, use the Disaster Recovery wizard.

1. In XenCenter select the secondary pool, and on the **Pool** menu, click **Disaster Recovery** to open the **Disaster Recovery** wizard.

   This wizard gives three recovery options: **Failover**, **Failback**, and **Test Failover**. To restore VMs and vApps to your primary site, select **Failback** and then click **Next**.

   > **Warning:**
   >
   > If you use Fibre Channel shared storage with LUN mirroring to replicate the data to the secondary site, before you attempt to recover data, **mirroring must be broken** so that the secondary site has Read/Write access.

2. Select the storage repositories (SRs) containing the pool metadata for the VMs and vApps that you want to recover.

   By default, the list on this wizard page shows all SRs that are currently attached within the pool. To scan for more SRs, choose **Find Storage Repositories** and then select the storage type to scan for:

   - To scan for all the available Hardware HBA SRs, select **Find Hardware HBA SRs**.
   - To scan for software iSCSI SRs, select **Find Software iSCSI SRs** and then enter the target host, IQN and LUN details in the dialog box.

   When you have selected the required SRs in the wizard, click **Next** to continue.

3. Select the VMs and vApps that you wish to restore and choose the appropriate **Power state after recovery** option to specify whether you want the wizard to start them up automatically as soon as they have been recovered, or whether you prefer to wait and start them up manually yourself after failback is complete.

   Click **Next** to progress to the next wizard page and begin failback prechecks.

4. The wizard performs a number of pre-checks before starting failback, for example, to ensure that all the storage required by the selected VMs and vApps is available. If any storage is missing at this point, you can click **Attach SR** on this page to find and attach the relevant SR.

   Resolve any issues on the prechecks page, and then click **Failback** to begin the recovery process.

5. A progress page is displayed showing whether recovery was successful for each VM and vApp. Failback may take some time depending on the number of VMs and vApps you are restoring, as the metadata for the VMs and vApps are exported from the replicated storage. The VMs and vApps are recreated in the primary pool, the SRs containing the virtual disks are attached to the recreated VMs, and VMs are started, if specified.

6. When the failback is complete, click **Next** to see the summary report. Click **Finish** on the summary report page to close the wizard.

Test Failover

Failover testing is an essential component in disaster recovery planning. You can use the Disaster Recovery wizard to perform non-disruptive testing of your disaster recovery system. During a test failover operation, all the steps are the same as for failover, but instead of being started after they have been recovered to the DR site, the VMs and vApps are placed in a paused state. At the end of a test failover operation, all VMs, vApps and storage recreated on the DR site are automatically removed. After initial DR configuration, and after you make significant configuration changes in a DR-enabled pool, we recommend that you verify that failover still works correctly by performing a test failover.

To perform a test failover of VMs and vApps to a secondary site

1. In XenCenter select the secondary pool, and on the **Pool** menu, click **Disaster Recovery** to open the **Disaster Recovery Wizard.**

2. Select **Test Failover** and then click **Next**.

   > **Note:**

If you use Fibre Channel shared storage with LUN mirroring to replicate the data to the secondary site, before you attempt to recover data, mirroring must be broken so that the secondary site has Read/Write access.

3. Select the storage repositories (SRs) containing the pool metadata for the VMs and vApps that you want to recover.

   By default, the list on this wizard page shows all SRs that are currently attached within the pool. To scan for more SRs, choose **Find Storage Repositories** and then select the storage type to scan for:

   - To scan for all the available Hardware HBA SRs, select **Find Hardware HBA SRs**.
   - To scan for software iSCSI SRs, select **Find Software iSCSI SRs** and then enter the target host, IQN and LUN details in the dialog box.

     When you have selected the required SRs in the wizard, click **Next** to continue.

4. Select the VMs and vApps that you wish to recover then click Next to progress to the next wizard page and begin failover prechecks.
5. Before beginning the test failover process, the wizard performs a number of pre-checks, for example, to ensure that all the storage required by the selected VMs and vApps is available.

   - **Check that storage is available** If any storage is missing, you can click **Attach SR** on this page to find and attach the relevant SR.
   - **Check that HA is not enabled on the target DR pool**. To avoid having the same VMs running on both the primary and DR pools, HA must be disabled on the secondary pool to ensure that the recovered VMs and vApps are not started up automatically by HA after recovery. To disable HA on the secondary pool, you can simply click **Disable HA** on the this page. (If HA is disabled at this point, it will be enabled again automatically at the end of the test failover process.)

     Resolve any issues on the pre-checks page, and then click **Failover** to begin the test failover.

6. A progress page is displayed showing whether recovery was successful for each VM and vApp. Failover may take some time depending on the number of VMs and vApps you are recovering, as the metadata for the VMs and vApps are recovered from the replicated storage. The VMs and vApps are recreated in the DR pool, the SRs containing the virtual disks are attached to the recreated VMs.

   The recovered VMs are placed in a paused state: they will not be started up on the secondary site during a test failover.

7. After you are satisfied that the test failover was performed successfully, click **Next** in the wizard to have the wizard clean up on the DR site:

   - VMs and vApps that were recovered during the test failover will be removed.
   - Storage that was recovered during the test failover will be detached.
   - If HA on the DR pool was disabled at the prechecks stage to allow the test failover to take place, it will be enabled again automatically.

     The progress of the cleanup process is displayed in the wizard.

8. Click **Finish** to close the wizard.

vApps

A **vApp** is logical group of one or more related Virtual Machines (VMs) which can be started up as a single entity in the event of a disaster. When a vApp is started, the VMs contained within the vApp will start in a user predefined order, to allow VMs which depend upon one another to be automatically sequenced. This means that an administrator no longer has to manually sequence the startup of dependant VMs should a whole service require restarting (for instance in the case of a software update). The VMs within the vApp do not have to reside on one host and will be distributed within a pool using the normal rules. The vApp functionality is particularly useful in the Disaster Recovery situation where an Administrator may choose to group all VMs which reside on the same Storage Repository, or which relate to the same Service Level Agreement (SLA).

Creating vApps

To group VMs together in a vApp follow the procedure:

1. Select the pool and, on the **Pool** menu, click **Manage vApps**. This displays the **Manage vApps** window.
2. Enter a name for the vApp, and optionally a description, and then click **Next**.

   You can choose any name you like, but a descriptive name is usually best. Although it is advisable to avoid having multiple vApps with the same name, it is not a requirement, and XenCenter does not enforce any uniqueness constraints on vApp names. It is not necessary to use quotation marks for names that include spaces.
3. Choose which VMs to include in the new vApp, and then click **Next**.

   You can use the search box to list only VMs with names that include the specified string.
4. Specify the startup sequence for the VMs in the vApp, and then click **Next**.

Value	Description
Start Order	Specifies the order in which individual VMs will be started up within the vApp, allowing certain VMs to be restarted before others. VMs with a start order value of 0 (zero) will be started first, then VMs with a start order value of 1, then VMs with a start order value of 2, and so on.
Attempt to start next VM after	This is a delay interval that specifies how long to wait after starting the VM before attempting to start the next group of VMs in the startup sequence, that is, VMs with a lower start order.

5. On the final page of the wizard, you can review the vApp configuration. Click **Previous** to go back and modify any settings, or **Finish** to create the new vApp and close the wizard.

   Note:

   A vApp can span multiple servers in a single pool, but cannot span across several pools.

Using the Manage vApps dialog box in XenCenter

XenCenter's **Manage vApps** dialog box allows you to create, delete and modify vApps, start and shutdown vApps, and import and export vApps within the selected pool. When you select a vApp in the list, the VMs it contains are listed in the details pane on the right. See the XenCenter online help for further details. Press **F1** or click **Help** to display the Help.

Backing Up and Restoring XenServer Hosts and VMs

Citrix recommends that, whenever possible, you leave the installed state of XenServer hosts unaltered. That is, do not install any additional packages or start additional services on XenServer hosts, and treat them as if they are appliances. The best way to restore, then, is to reinstall XenServer host software from the installation media. If you have multiple XenServer hosts, the best approach is to configure a PXE boot server and appropriate answerfiles for this purpose (see the *XenServer Installation Guide*).

For VMs, the best approach is to install backup agents on them, just as if they were standard physical servers. For Windows VMs, as of this release we have tested CA BrightStor ARCserve Backup, and Symantec NetBackup and Backup Exec.

For more information about backup tools tested, best practices, and backups in general, see the Citrix Knowledge Base.

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

To backup pool metadata

1. Run the command:

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

2. Run the command:

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

Backing up Virtual Machine metadata

XenServer hosts use a database on each host to store metadata about VMs and associated resources such as storage and networking. When combined with storage repositories, this database forms the complete view of all VMs available across the pool. Therefore it is important to understand how to backup this database in order to recover from physical hardware failure and other disaster scenarios.

This section first describes how to backup metadata for single-host installations, and then for more complex pool setups.

Backing up single host installations

Use the CLI to backup the pool database. To obtain a consistent pool metadata backup file, run **pool-dump-database** on the XenServer host and archive the resulting file. The backup file will contain sensitive authentication information about the pool, so ensure it is securely stored.

To restore the pool database, use the **xe pool-restore-database** command from a previous dump file. If your XenServer host has died completely, then you must first do a fresh install, and then run the **pool-restore-database** command against the freshly installed XenServer host.

After a restoration of the pool database, some VMs may still be registered as being `Suspended`, but if the storage repository with their suspended memory state (defined in the *`suspend-VDI-uuid`* field) was a local SR, it will no longer be available since the host has been reinstalled. To reset these VMs back to the `Halted` state so that they can be started up again, use the **xe vm-shutdown vm=vm_name -force** command, or use the **xe vm-reset-powerstate vm=<*vm_name*> -force** command.

> **Warning:**
>
> XenServer hosts restored using this method will have their UUIDs preserved. If you restore to a different physical machine while the original XenServer host is still running, there will be duplicate UUIDs. The main observable effect of this will be that XenCenter will refuse to connect to the second XenServer host. Pool database backup is not the recommended mechanism for cloning physical hosts; use the automated installation support for that (see the *XenServer Installation Guide*).

Backing up pooled installations

In a pool scenario, the master host provides an authoritative database that is synchronously mirrored to all the member hosts in the pool. This provides a degree of built-in redundancy to a pool; the master can be replaced by any member since each of them has an accurate version of the pool database. Please refer to the *XenServer Administrator's Guide* for more information on how to transition a member into becoming a master host.

This level of protection may not be sufficient; for example, if your shared storage containing the VM data is backed up in multiple sites, but your local server storage (containing the pool metadata) is not. To fully recreate a pool given just a set of shared storage, you must first backup the **pool-dump-database** file on the master host, and archive this file.

Subsequently restoring this backup on a brand new set of hosts

1. Install a fresh set of XenServer hosts from the installation media, or over PXE.
2. Use the **xe pool-restore-database** on the host designated to be the new master.
3. Run the **xe host-forget** command on the new master to remove the old member machines.
4. Use the **xe pool-join** command on the member hosts to connect them to the new pool.

Backing up XenServer hosts

This section describes the XenServer host control domain backup and restore procedures. These procedures do *not* back up the storage repositories that house the VMs, but only the privileged control domain that runs Xen and the XenServer agent.

> **Note:**
>
> Because the privileged control domain is best left as installed, without customizing it with other packages, Citrix recommends that you set up a PXE boot environment to cleanly perform a fresh installation from the XenServer media as a recovery strategy. In many cases you will not need to backup the control domain at all, but just save the pool metadata (see the

section called “Backing up Virtual Machine metadata”). This backup method should always be considered complementary to backing up the pool metadata.

Using the xe commands **host-backup** and **host-restore** is another approach that you can take. The xe **host-backup** command archives the active partition to a file you specify, and the xe **host-restore** command extracts an archive created by xe **host-backup** over the currently inactive disk partition of the host. This partition can then be made active by booting off the installation CD and choosing to restore the appropriate backup.

After completing the above steps and rebooting the host, you must ensure that the VM meta-data is restored to a consistent state. This can be achieved by running **xe pool-restore-database** on `/var/backup/pool-database-${DATE}`. This file is created by **xe host-backup** using **xe pool-dump-database** command before archiving the running filesystem, to snapshot a consistent state of the VM metadata.

To back up a XenServer host

- On a remote host with enough disk space, run the command:

  ```
  xe host-backup file-name=<filename> -h <hostname> -u root -pw <password>
  ```

  This creates a compressed image of the control domain file system in the location specified by the *`file-name`* argument.

To restore a running XenServer host

1. If you want to restore a XenServer host from a specific backup, run the following command while the XenServer host is up and reachable:

   ```
   xe host-restore file-name=<filename> -h <hostname> -u root -pw <password>;
   ```

   This command restores the compressed image back to the hard disk of the XenServer host on which the command is run (not the host on which *`filename`* resides). In this context “restore” is something of a misnomer, as the word usually suggests that the backed-up state has been put fully in place. The restore command here only unpacks the compressed backup file and restores it to its normal form, but it is written to another partition (`/dev/sda2`) and does *not* overwrite the current version of the filesystem.
2. To use the restored version of the root filesystem, reboot the XenServer host using the XenServer installation CD and select the **Restore from backup** option.

   After the restore from backup is completed, reboot the XenServer host and it will start up from the restored image.

   Finally, restore the VM meta-data using

   ```
   xe pool-restore-database file-name=/var/backup/pool-database-*
   ```

   > **Note:**
   >
   > Restoring from a backup as described here does *not* destroy the backup partition.

Restarting a crashed XenServer host

1. If your XenServer host has crashed and is not reachable anymore, you need to use the XenServer installation CD to do an upgrade install. When that is completed, reboot the machine and make sure your host is reachable with XenCenter or remote CLI.
2. Then proceed with the section called “Backing up XenServer hosts” above.

Backing up VMs

VMs are best backed up using standard backup tools running on them individually. For Windows VMs, we have tested CA BrightStor ARCserve Backup.

VM Snapshots

XenServer 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. Snapshots are 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 XenServer and the volume must be in the default format (`type=raw` volumes cannot be snapshotted).

The snapshotting operation is a 2 step process:

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

Three types of VM snapshots are supported: regular, quiesced, and snapshot with memory

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.

XenServer 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. For further detail on quiesced snapshots, see the section called “Advanced Notes for Quiesced Snapshots”.

Snapshots with memory

In addition to saving the VMs memory (storage) and metadata, snapshots with memory also save the VMs state (RAM). This can be useful if you are upgrading or patching software, or want to test a new application, but also want the option to be able to get back to the current, pre-change state (RAM) of the VM. Reverting back to a snapshot with memory, does not require a reboot of the VM.

You can take a snapshot with memory of a running or suspended VM via the XenAPI, the xe CLI, or by using XenCenter.

Creating a VM Snapshot

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

Firstly, ensure that the VM is running or suspended so that the memory status can be captured. The simplest way to select the VM on which the operation is to be performed is by supplying the argument `vm=`*`<name>`* or `vm=`*`<vm uuid>>`* .

Run the **vm-snapshot** and **vm-snapshot-with-quiesce** commands to take a snapshot of a VM.

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

Creating a snapshot with memory

Run the **vm-checkpoint** command, giving a descriptive name for the snapshot with memory, so that you can identify it later:

```
xe vm-checkpoint vm=<vm uuid> new-name-label=<name of the checkpoint>
```

When XenServer has completed creating the snapshot with memory - its uuid will be displayed.

For example:

```
xe vm-checkpoint vm=2d1d9a08-e479-2f0a-69e7-24a0e062dd35 \
   new-name-label=example_checkpoint_1
b3c0f369-59a1-dd16-ecd4-a1211df29886
```

A snapshot with memory requires at least 4MB of disk space per disk, plus the size of the RAM, plus around 20% overhead. So a checkpoint with 256MB RAM would require approximately 300MB of storage.

> **Note:**
>
> During the checkpoint creation process, the VM is paused for a brief period of time, and cannot be used during this period.

To list all of the snapshots on a XenServer pool

Run the **snapshot-list** command:

```
xe snapshot-list
```

This lists all of the snapshots in the XenServer pool.

To list the snapshots on a particular VM

You will need to know the uuid of the particular VM; to do this run the **vm-list** command.

```
xe vm-list
```

This displays a list of all VMs and their UUIDs. For example:

```
xe vm-list
uuid ( RO): 116dd310-a0ef-a830-37c8-df41521ff72d
name-label ( RW): Windows Server 2003 (1)
power-state ( RO): halted

uuid ( RO): 96fde888-2a18-c042-491a-014e22b07839
name-label ( RW): Windows XP SP3 (1)
power-state ( RO): running

uuid ( RO): dff45c56-426a-4450-a094-d3bba0a2ba3f
name-label ( RW): Control domain on host
power-state ( RO): running
```

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 field is equal to 'halted'. Where multiple VMs are matching, the option **--multiple** must be specified to perform the operation. The full list of fields that can be matched can be obtained by the command **xe vm-list params=all**.

Locate the required VM and then enter the following:

```
xe snapshot-list snapshot-of=<vm uuid>
```

For example:

```
xe snapshot-list snapshot-of=2d1d9a08-e479-2f0a-69e7-24a0e062dd35
```

This lists the snapshots currently on that VM:

```
        uuid ( RO): d7eefb03-39bc-80f8-8d73-2ca1bab7dcff
        name-label ( RW): Regular
        name-description ( RW):
        snapshot_of ( RO): 2d1d9a08-e479-2f0a-69e7-24a0e062dd35
        snapshot_time ( RO): 20090914T15:37:00Z

        uuid ( RO): 1760561d-a5d1-5d5e-2be5-d0dd99a3b1ef
        name-label ( RW): Snapshot with memory
        name-description ( RW):
        snapshot_of ( RO): 2d1d9a08-e479-2f0a-69e7-24a0e062dd35
        snapshot_time ( RO): 20090914T15:39:45Z
```

Restoring a VM to its previous state

Ensure that you have the uuid of the snapshot that you want to revert to, and then run the **snapshot-revert** command:

To do this:

1. Run the **snapshot-list** command to find the UUID of the snapshot or checkpoint that you want to revert to:

   ```
   xe snapshot-list
   ```

2. Note the uuid of the snapshot, and then run the following command to revert:

   ```
   xe snapshot-revert snapshot-uuid=<snapshot uuid>
   ```

   For example:

   ```
   xe snapshot-revert snapshot-uuid=b3c0f369-59a1-dd16-ecd4-a1211df29886
   ```

   After reverting to a checkpoint, the VM will be suspended.

   > **Note:**
   >
   > It is possible to revert to any snapshot in time,forwards or backwards. Existing snapshots and checkpoints are not deleted during revert.

Deleting a snapshot

Ensure that you have the UUID of the checkpoint or snapshot that you wish to remove, and then run the following command:

1. Run the **snapshot-list** command to find the UUID of the snapshot or checkpoint that you want to revert to:

   ```
   xe snapshot-list
   ```

2. Note the UUID of the snapshot, and then run the **snapshot-uninstall** command to remove it:

   ```
   xe snapshot-uninstall snapshot-uuid=<snapshot-uuid>
   ```

3. This command alerts you to the VM and VDIs that will be deleted. Type **yes** to confirm.

For example:

```
xe snapshot-uninstall snapshot-uuid=1760561d-a5d1-5d5e-2be5-d0dd99a3b1ef
The following items are about to be destroyed
VM : 1760561d-a5d1-5d5e-2be5-d0dd99a3b1ef (Snapshot with memory)
VDI: 11a4aa81-3c6b-4f7d-805a-b6ea02947582 (0)
VDI: 43c33fe7-a768-4612-bf8c-c385e2c657ed (1)
VDI: 4c33c84a-a874-42db-85b5-5e29174fa9b2 (Suspend image)
Type 'yes' to continue
yes
All objects destroyed
```

If you only want to remove the metadata of a checkpoint or snapshot, run the following command:

```
xe snapshot-destroy snapshot-uuid=<snapshot-uuid>
```

For example:

```
xe snapshot-destroy snapshot-uuid=d7eefb03-39bc-80f8-8d73-2ca1bab7dcff
```

Snapshot Templates

Creating a template from a snapshot

You can create a VM template from a snapshot, however its memory state will be removed.

To do this:

1. Use the command**snapshot-copy** and specify a **new-name-label** for the template:

   ```
   xe snapshot-copy new-name-label=<vm-template-name> \
     snapshot-uuid=<uuid of the snapshot>
   ```

   For example:

   ```
   xe snapshot-copy new-name-label=example_template_1
     snapshot-uuid=b3c0f369-59a1-dd16-ecd4-a1211df29886
   ```

   Note:

   This creates a template object in the SAME pool. This template exists in the XenServer database for the current pool only.

2. To verify that the template has been created, run the command **template-list**:

   ```
   xe template-list
   ```

   This will list all of the templates on the XenServer host.

Exporting a snapshot to a template

When you export a VM snapshot, a complete copy of the VM (including disk images) is stored as a single file on your local machine, with a **.xva** file extension.

To do this:

1. Use the command **snapshot-export-to-template** to create a new template file:

   ```
   xe snapshot-export-to template snapshot-uuid=<snapshot-uuid> \
     filename=<template- filename>
   ```

   For example:

   ```
   xe snapshot-export-to-template snapshot-uuid=b3c0f369-59a1-dd16-ecd4-a1211df29886 \
     filename=example_template_export
   ```

The VM export/import feature can be used in a number of different ways:

- As a convenient backup facility for your VMs. An exported VM file can be used to recover an entire VM in the event of disaster.
- As a way of quickly copying a VM, for example, a special-purpose server configuration that you use many times. You simply configure the VM the way you want it, export it, and then import it to create copies of your original VM.
- As a simple method for moving a VM to another server.

For further information on the use of templates refer to the Creating VMs chapter in the XenServer Virtual Machine Installation Guide and also the Managing virtual machines section in the XenCenter Help.

Advanced Notes for Quiesced Snapshots

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 XenServer administrator whereby adding an attribute of `snapmanager=true` to the VM `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. Citrix 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 XenServer 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 XenServer 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 XenServer VSS hardware provider do not support snapshots of volumes hosted on dynamic disks on the Windows VM.

Note:

Using EqualLogic or NetApp storage requires a **Citrix XenServer Advanced Edition or higher** license. To learn more about XenServer editions, and to find out how to upgrade, visit the Citrix website here.

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.

VM Protection and Recovery

Note:

VM Protection and Recovery requires a Citrix XenServer Platinum Edition. To learn more about XenServer editions and to find out how to upgrade, visit the Citrix web site here.

XenServer's VM Protection and Recovery (VMPR) feature provides a simple backup and restore utility for your critical service VMs. Regular scheduled snapshots are taken automatically and can be used to restore VMs in case of disaster. Scheduled snapshots can also be automatically archived to a remote CIFS or NFS share, providing an additional level of security.

VM Protection and Recovery works by having pool-wide VM protection policies that define snapshot schedules for selected VMs in the pool. When a policy is enabled, snapshots are taken of the specified VMs at the scheduled time each hour, day or week, and, if configured, the snapshots can also be archived automatically. Several policies may be enabled in a pool, covering different VMs and with different schedules.

Note:

A VM can only be assigned to one VMPR policy at a time.

Naming convention for VM archive folders

When using VMPR, the following naming convention for archive folders and their contents applies. The archive folder name combines the VM name and the first 16 characters of the VM UUID. For example:

If the VM name = `Win7_Test_1`, and the VM UUID = `cb53200c-bbd8-4c12-a076-e2eb29b38f06` then the archive folder is named `Win7_Test_1-cb53200c-bbd8-4c`.

This folder contains archived VM files, in the `YYYYMMDD-HHMM.xva` form. For example:

```
20100624-1830.xva
```

```
20100625-1830.xva
```

```
20100625-1830.xva
```

For more information on using VM Protection and Recovery, please see the XenCenter online help.

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 “Backing Up and Restoring XenServer Hosts and VMs”.

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 600 seconds, the master assumes the member is dead. There are two ways to recover from this problem:

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

> **Warning:**
>
> - If you are going to use the forgotten host as a XenServer host again, perform a fresh installation of the XenServer 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 XenServer host fails, there may be VMs still registered in the *running* state. If you are sure that the member XenServer host is definitely down, 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.

Before you can start VMs on another XenServer host, you are also required to release the locks on VM storage. Each disk in an SR can only be used by one host at a time, so it is key to make the disk accessible to other XenServer hosts once a host has failed. To do so, run the following script on the pool master for each SR that contains disks of any affected VMs:

`/opt/xensource/sm/resetvdis.py` *<host_UUID> <SR_UUID>* [master]

You need only supply the third string ("master") if the failed host was the SR master — pool master or XenServer host using local storage — at the time of the crash.

> **Warning:**
>
> Be absolutely sure that the host is down before executing this command. Incorrect use of this command can lead to data corruption.

If you attempt to start a VM on another XenServer host before running the script above, then you will receive the following error message: `VDI <UUID> already attached RW`.

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. If HA is enabled, another master is elected automatically.
2. If HA is not enabled, each member will wait for the master to return.

If the master comes back up at this point, it re-establishes communication with its members, 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 XenServer host software, and add it to the pool. Since the XenServer hosts in the pool are enforced to be homogeneous, there is no real need to make the replaced server the master.

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

Pool failures

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

To restore a completely failed pool

1. Install a fresh set of hosts. 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. Connect to the master host using XenCenter and ensure that all your shared storage and VMs are available again.
4. Perform a pool join operation on the remaining freshly installed member hosts, and start up your VMs on the appropriate hosts.

Coping with Failure due to Configuration Errors

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

To restore host software and configuration

1. Run the command:

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

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

Physical Machine failure

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

> **Warning:**
>
> Any VMs which were running on a previous member (or the previous host) which has failed will still be marked as `Running` in the database. This is for safety-- simultaneously starting a VM on two different hosts would lead to severe disk corruption. If you are sure that the machines (and VMs) are offline you can reset the VM power state to `Halted`:
>
> ```
> xe vm-reset-powerstate vm=<vm_uuid> --force
> ```
>
> VMs can then be restarted using XenCenter or the CLI.

Replacing a failed master with a still running member

1. Run the commands:

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

2. If the commands succeed, restart the VMs.

To restore a pool with all hosts failed

1. Run the command:

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

   Warning:

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

2. If the target machine has a different view of the storage (for example, a block-mirror with a different IP address) than the original machine, modify the storage configuration using 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** or **Storage > Repair Storage Repository** menu item in XenCenter 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=true
   ```

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

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

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

Monitoring and Managing XenServer

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

> **Note:**
>
> Full monitoring and alerting functionality is only available with XenServer Advanced edition or above. To find out about XenServer editions and how to upgrade, visit the Citrix website here.

Alerts

XenServer generates alerts for the following events.

Configurable Alerts:

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

Alerts generated by XenCenter:

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

Alerts generated by XenServer:

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

The following alerts appear on the performance graphs in XenCenter. See the *XenCenter online help* for more information:

- vm_cloned
- vm_crashed
- vm_rebooted
- vm_resumed
- vm_shutdown
- vm_started
- vm_suspended

Customizing Alerts

> **Note:**
>
> Triggers for alerts are checked at a minimum interval of five minutes (this avoids placing excessive load on the system to check for these conditions and reporting of false positives); setting an alert repeat interval smaller than this will result in the alerts still being generated at the five minute minimum interval.

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

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

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

```
<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

> **Note:**
>
> Email alerts are only available in a pool with XenServer Advanced edition or above. To find out about XenServer editions and how to upgrade, visit the Citrix website here.

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

```
pool:other-config:mail-destination=<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.

Custom Fields and Tags

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

Custom Searches

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

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.

Troubleshooting

Citrix provides two forms of support: free, self-help support on the Citrix Support website and paid-for Support Services, which you can purchase from the Support site. With Citrix Technical Support, you can open a Support Case online or contact the support center by phone if you experience technical difficulties.

The Citrix Knowledge Center hosts a number of resources that may be helpful to you if you experience odd behavior, crashes, or other problems. Resources include: Support Forums, Knowledge Base articles and product documentation.

If you experience technical difficulties with a XenServer host, this chapter is meant to help you solve the problem if possible and, failing that, describes where the application logs are located and other information that can help your Citrix Solution Provider and Citrix track and resolve the issue.

Troubleshooting of installation issues is covered in the *XenServer Installation Guide*. Troubleshooting of Virtual Machine issues is covered in the *XenServer Virtual Machine Installation Guide*.

> **Important:**
>
> We recommend that you follow the troubleshooting information in this chapter solely under the guidance of your Citrix Solution Provider or Citrix Support.

XenServer host logs

XenCenter can be used to gather XenServer host information. Click on **Get Server Status Report...** in the **Tools** menu to open the **Server Status Report** wizard. You can select from a list of different types of information (various logs, crash dumps, etc.). The information is compiled and downloaded to the machine that XenCenter is running on. For details, see the XenCenter Help.

Additionally, the XenServer host has several CLI commands to make it simple to collate the output of logs and various other bits of system information using the utility **xen-bugtool**. Use the xe command **host-bugreport-upload** to collect the appropriate log files and system information and upload them to the Citrix Support ftp site. Please refer to the section called “host-bugreport-upload” for a full description of this command and its optional parameters. If you are requested to send a crashdump to Citrix Support, use the xe command **host-crashdump-upload**. Please refer to the section called “host-crashdump-upload” for a full description of this command and its optional parameters.

> It is possible that sensitive information might be written into the XenServer host logs.

By default, the server logs report only errors and warnings. If you need to see more detailed information, you can enable more verbose logging. To do so, use the **host-loglevel-set** command:

host-loglevel-set *`log-level=level`*

where *`level`* can be 0, 1, 2, 3, or 4, where 0 is the most verbose and 4 is the least verbose.

The default setting is to keep 20 rotations of each file, and the **logrotate** command is run daily.

Sending host log messages to a central server

Rather than have logs written to the control domain filesystem, you can configure a XenServer host to write them to a remote server. The remote server must have the syslogd daemon running on it to receive the logs and aggregate them correctly. The syslogd daemon is a standard part of all flavors of Linux and Unix, and third-party versions are available for Windows and other operating systems.

To write logs to a remote server

1. Set the syslog_destination parameter to the hostname or IP address of the remote server where you want the logs to be written:

```
xe host-param-set uuid=<xenserver_host_uuid> logging:syslog_destination=<hostname>
```

2. Issue the command:

   ```
   xe host-syslog-reconfigure uuid=<xenserver_host_uuid>
   ```

   to enforce the change. (You can also execute this command remotely by specifying the `host` parameter.)

XenCenter logs

XenCenter also has client-side log. This file includes a complete description of all operations and errors that occur when using XenCenter. It also contains informational logging of events that provide you with an audit trail of various actions that have occurred. The XenCenter log file is stored in your profile folder. If XenCenter is installed on Windows XP, the path is

```
%userprofile%\AppData\Citrix\XenCenter\logs\XenCenter.log
```

If XenCenter is installed on Windows Vista, the path is

```
%userprofile%\AppData\Citrix\Roaming\XenCenter\logs\XenCenter.log
```

To quickly locate the XenCenter log files, for example, when you want to open or email the log file, click on **View Application Log Files** in the XenCenter **Help** menu.

Troubleshooting connections between XenCenter and the XenServer host

If you have trouble connecting to the XenServer host with XenCenter, check the following:

- Is your XenCenter an older version than the XenServer host you are attempting to connect to?

  The XenCenter application is backward-compatible and can communicate properly with older XenServer hosts, but an older XenCenter cannot communicate properly with newer XenServer hosts.

  To correct this issue, install a XenCenter version that is the same, or newer, than the XenServer host version.
- Is your license current?

  You can see the expiration date for your License Key in the XenServer host **General** tab under the **Licenses** section in XenCenter.

  Also, if you upgraded your software from version 3.2.0 to the current version, you should also have received and applied a new License file.

  For details on licensing a host, see the chapter "XenServer Licensing" in the *XenServer Installation Guide*.
- The XenServer host talks to XenCenter using HTTPS over port 443 (a two-way connection for commands and responses using the XenAPI), and 5900 for graphical VNC connections with paravirtual Linux VMs. If you have a firewall enabled between the XenServer host and the machine running the client software, make sure that it allows traffic from these ports.

Appendix A. Command Line Interface

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

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

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

To use it, open a Windows Command Prompt and change directories to the directory where the file resides (typically `C:\Program Files\Citrix\XenCenter`), or add its installation location to your system path.

On RPM-based distributions (such as, Red Hat and CentOS), you can install the stand-alone xe CLI executable from the RPM named `xe-cli-6.00-@BUILD_NUMBER@.i386.rpm` on the main XenServer installation ISO, as follows:

```
rpm -ivh xe-cli-6.00-@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 XenServer xe CLI commands is:

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

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

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

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

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

Example: On the local XenServer host:

```
xe vm-list
```

Example: On the remote XenServer host:

```
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 XenServer host:

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

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

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

This command means that you will not need to specify the remote XenServer 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 XenServer 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 XenServer host, commands have a tab completion feature similar to that in the standard Linux bash shell. If you type, for example**xe vm-l** and then press the **TAB** key, the rest of the command will be displayed when it is unambiguous. If more than one command begins with vm-l, 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 XenServer host, tab completion does not normally work. However if you put the server, username, and password in an environment variable called XE_EXTRA_ARGS on the machine from which you are entering the commands, tab completion is enabled. See the section called “Basic xe Syntax” for details.

Command Types

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

- *<class>*-list

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

where *<class>* is one of:

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

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 Parameter 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. For example:

```
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 XenServer host with three VMs installed gives the three UUIDs of the VMs, for example:

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

xe Command Reference

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

Appliance Commands

Commands for creating and modifying VM appliances (also known as vApps). For more information on vApps, see the XenServer Virtual Machine Installation Guide.

Appliance Parameters

Appliance commands have the following parameters

Parameter Name	Description	Type
uuid	appliance uuid	required
name-description	appliance description	optional
paused		optional
force	force shutdown	optional

appliance-assert-can-be-recovered

`appliance-assert-can-be-recovered` uuid=*<appliance-uuid>* database:vdi-uuid=*<vdi-uuid>*

Tests whether storage is available to recover this VM appliance/vApp.

appliance-create

`appliance-create` name-label=*<name-label>* [name-description=*<name-description>*]

Creates an appliance/vApp. For example:

```
xe appliance-create name-label=my_appliance
```

Add VMs to the appliance:

```
xe vm-param-set uuid=<VM-UUID> appliance=<appliance-uuid> \
xe vm-param-set uuid=<VM-UUID> appliance=<appliance-uuid>
```

appliance-destroy

`appliance-destroy` uuid=*<appliance-uuid>*

Destroys an appliance/vApp. For example:

```
xe appliance-destroy uuid=<appliance-uuid>
```

appliance-recover

`appliance-recover` uuid=*<appliance-uuid>* database:vdi-uuid=*<vdi-uuid>* [paused=*<true|false>*]

Recover a VM appliance/vAPP from the database contained in the supplied VDI.

appliance-shutdown

`appliance-shutdown` uuid=*<appliance-uuid>* [force=*<true|false>*]

Shuts down all VMs in an appliance/vApp. For example:

```
xe appliance-shutdown uuid=<appliance-uuid>
```

appliance-start

`appliance-start` uuid=*<appliance-uuid>* [paused=*<true|false>*]

Starts an appliance/vApp. For example:

```
xe appliance-start uuid=<appliance-uuid>
```

Audit Commands

Audit commands download all of the available records of the RBAC audit file in the pool. If the optional parameter `since` is present, it downloads only the records from that specific point in time.

audit-log-get parameters

audit-log-get has the following parameters

Parameter Name	Description	Type
filename	Write the audit log of the pool to *<filename>*	required
since	specific date/time point	optional

audit-log-get

`audit-log-get` [since=*<timestamp>*] filename=*<filename>*

For example, to obtain audit records of the pool since a precise millisecond timestamp, run the following command:

Run the following command:

```
xe audit-log-get since=2009-09-24T17:56:20.530Z \
filename=/tmp/auditlog-pool-actions.out
```

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
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 XenServer host, or if fewer than 2 PIFs are supplied.

bond-destroy

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

Deletes a bonded interface specified by its UUID from a XenServer host.

CD Commands

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

CD Parameters

CDs have the following parameters:

Parameter Name	Description	Type
uuid	unique identifier/object reference for the CD	read only
name-label	Name for the CD	read/write
name-description	Description text for the CD	read/write
allowed-operations	A list of the operations that can be performed on this CD	read only set parameter
current-operations	A list of the operations that are currently in progress on this CD	read only set parameter

Parameter Name	Description	Type
sr-uuid	The unique identifier/object reference for the SR this CD is part of	read only
sr-name-label	The name for the SR this CD is part of	read only
vbd-uuids	A list of the unique identifiers for the VBDs on VMs that connect to this CD	read only set parameter
crashdump-uuids	Not used on CDs since crashdumps cannot be written to them	read only set parameter
virtual-size	Size of the CD as it appears to VMs (in bytes)	read only
physical-Utilization	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 writable. 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
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 XenServer host or pool, filtering on the optional argument *params*.

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

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

Console Commands

Commands for working with consoles.

The console objects can be listed with the standard object listing command (**xe console-list**), and the parameters manipulated with the standard parameter commands. See the section called "Low-level Parameter Commands" for details.

Console Parameters

Consoles have the following parameters:

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

Disaster Recovery (DR) Commands

Commands for recovering VMs in the event of disaster

drtask-create

drtask-create type=*<type>* sr-whitelist=*<sr-white-list>* device-config=*<device-config>*

Creates a disaster recovery task. For example, to connect to an iSCSI SR in preparation for Disaster Recovery:

```
xe dr-task-create type=lvmoiscsi device-config:target=<target-ip-address> \
  device-config:targetIQN=<targetIQN> device-config:SCSIid=<SCSIid> \
  sr-whitelist=<sr-uuid-list>
```

Note:

sr-whitelist lists SR UUIDs, **drtask-create** will only introduce and connect to an SR which has one of the whitelisted UUIDs

drtask-destroy

`drtask-destroy` uuid=*<dr-task-uuid>*

Destroys a disaster recovery task and forgets the introduced SR.

vm-assert-can-be-recovered

`vm-assert-can-be-recovered` uuid=*<vm-uuid>* database:vdi-uuid=*<vdi-uuid>*

Tests whether storage is available to recover this VM.

appliance-assert-can-be-recovered

`appliance-assert-can-be-recovered` uuid=*<appliance-uuid>* database:vdi-uuid=*<vdi-uuid>*

Checks whether the storage (containing the appliance's/vAPP disk) is visible.

appliance-recover

`appliance-recover` uuid=*<appliance-uuid>* database:vdi-uuid=*<vdi-uuid>* [force=*<true|false>*]

Recover an appliance/vAPP from the database contained in the supplied VDI.

vm-recover

`vm-recover` uuid=*<vm-uuid>* database:vdi-uuid=*<vdi-uuid>* [force=*<true|false>*]

Recovers a VM from the database contained in the supplied VDI.

sr-enable-database-replication

`sr-enable-database-replication` uuid=*<sr_uuid>*

Enables xapi database replication to the specified (shared) SR. For example:

```
xe sr-enable-database-replication uuid=<sr-uuid>
```

sr-disable-database-replication

`sr-disable-database-replication` uuid=*<sr_uuid>*

Disables xapi database replication to the specified SR. For example:

```
xe sr-enable-database-replication uuid=<sr-uuid>
```

Example Usage

The example below shows the DR CLI commands in context:

On the primary site, enable database replication:

```
xe sr-database-replication uuid=<sr=uuid>
```

In the event of disaster, on the secondary site, connect to the SR:(note **device-config** has the same fields as **sr-probe**).

```
xe drtask-create type=lvmoiscsi \
   device-config:target=<target ip address> \
   device-config:targetIQN=<target-iqn> \
   device-config:SCSIid=<scsi-id> \
   sr-whitelist=<sr-uuid>
```

Look for database VDIs on the SR:

```
xe vdi-list sr-uuid=<sr-uuid> type=Metadata
```

Query a database VDI for VMs present:

```
xe vm-list database:vdi-uuid=<vdi-uuid>
```

Recover a VM:

```
xe vm-recover uuid=<vm-uuid> database:vdi-uuid=<vdi-uuid>
```

Destroy the DR task; any SRs introduced by the DR task and not required by VMs are destroyed:

```
xe drtask-destroy uuid=<drtask-uuid>
```

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

Class name	Description
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.

GPU Commands

Commands for working with physical GPUs, GPU groups and virtual GPUs.

The GPU objects can be listed with the standard object listing commands (**xe pgpu-list**, **xe gpu-group-list**, and **xe vgpu-list**), and the parameters manipulated with the standard parameter commands. For details, see the section called "Low-level Parameter Commands".

Physical GPU (pGPU) Parameters

pGPUs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the pGPU	Read only
vendor-name	The vendor name of the pGPU	Read only
device-name	The name assigned by the vendor to this pGPU model	Read only
gpu-group-uuid	The unique identifier/object reference for the GPU group that this pGPU has been automatically assigned to by XenServer; identical pGPUs across hosts in a pool are grouped together	Read only

Parameter Name	Description	Type
gpu-group-name-label	The name of the GPU group to which the pGPU is assigned	Read only
host-uuid	The unique identifier/object reference for the XenServer host to which the pGPU is connected	Read only
host-name-label	The name of the XenServer host to which the pGPU is connected	Read only
pci-id	PCI identifier	Read only
dependencies	Lists the dependent PCI devices passed-through to the same VM	Read/write map parameter
other-config	A list of key/value pairs that specify additional configuration parameters for the pGPU	Read/write map parameter

GPU Group Parameters

GPU groups have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the GPU group	Read only
name-label	The name of the GPU group	Read/write
name-description	The descriptive text of the GPU group	Read/write
VGPU-uuids	Lists the unique identifier/ object references for the vGPUs in the GPU group	Read only set parameter
PGPU-uuids	Lists the unique identifier/ object references for the pGPUs in the GPU group	Read only set parameter
other-config	A list of key/value pairs that specify additional configuration parameters for the GPU group	Read/write map parameter

Virtual GPU (vGPU) Parameters

vGPUs have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the vGPU	Read only

Parameter Name	Description	Type
vm-uuid	The unique identifier/object reference for the VM to which the vGPU is assigned	Read only
vm-name-label	The name of the VM to which the vGPU is assigned	Read only
gpu-group-uuid	The unique identifier/object reference for the GPU group in which the vGPU is contained	Read only
gpu-group-name-label	The name of the GPU group in which the vGPU is contained	Read only
currently-attached	`True` if a VM with GPU Pass-Through is running, `false` otherwise	Read only
other-config	A list of key/value pairs that specify additional configuration parameters for the vGPU	Read/write map parameter

vgpu-create

`vgpu-create` vm-uuid=*<uuid_of_vm>* gpu_group_uuid=*<uuid_of_gpu_group>*

Create a vGPU. This command attaches the VM to the specified GPU group.

vgpu-destroy

`vgpu-destroy` uuid=*<uuid_of_vgpu>*

Destroy a vGPU. This command detaches the VM from the associated GPU group.

Host Commands

Commands for interacting with XenServer host.

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

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

Host Selectors

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

Host Parameters

XenServer hosts have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the XenServer host	read only
name-label	The name of the XenServer host	read/write
name-description	The description string of the XenServer host	read only
enabled	false if disabled which prevents any new VMs from starting on them, which prepares the XenServer hosts to be shut down or rebooted; true if the host is currently enabled	read only
API-version-major	major version number	read only
API-version-minor	minor version number	read only
API-version-vendor	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 XenServer host can run	read only set parameter
other-config	A list of key/value pairs that specify additional configuration parameters for the XenServer host	read/write map parameter
chipset-info	A list of key/value pairs that specify information about the chipset	Read only map parameter
hostname	XenServer host hostname	read only
address	XenServer host IP address	read only

Parameter Name	Description	Type
license-server	• A list of key/value pairs that specify information about the license server • The default port for communications with Citrix products is 27000. For information on changing port numbers due to conflicts, see the Licensing Your Product section on Citrix eDocs.	Read only map parameter
supported-bootloaders	list of bootloaders that the XenServer host supports, for example, pygrub, eliloader	read only set parameter
memory-total	total amount of physical RAM on the XenServer host, in bytes	read only
memory-free	total amount of physical RAM remaining that can be allocated to VMs, in bytes	read only
host-metrics-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
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

Parameter Name	Description	Type
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

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

CPUs on XenServer hosts have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the CPU	read only
number	the number of the physical CPU core within the XenServer host	read only
vendor	the vendor string for the CPU name, for example, "GenuineIntel"	read only
speed	The CPU clock speed, in Hz	read only
modelname	the vendor string for the CPU model, for example, "Intel(R) Xeon(TM) CPU 3.00GHz"	read only
stepping	the CPU revision number	read only
flags	the flags of the physical CPU (a decoded version of the features field)	read only
Utilization	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 XenServer hosts have the following parameters:

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

Parameter Name	Description	Type
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 XenServer host to the machine that the command is invoked from, and save it there as a file with the name *`file-name`*.

> 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 Citrix 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 parameters listed at the beginning of this section.

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

host-crashdump-destroy

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

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

host-crashdump-upload

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

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

host-disable

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

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

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

host-dmesg

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

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

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

host-emergency-management-reconfigure

`host-emergency-management-reconfigure` interface=*<uuid_of_management_interface_pif>*

Reconfigure the primary management interface of this XenServer host. Use this command only if the XenServer 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 XenServer 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 parameters 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 parameters listed at the beginning of this section.

host-forget

`host-forget` uuid=*<XenServer_host_UUID>*

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

Use the `--force` parameter to avoid being prompted to confirm that you really want to perform this operation.

> **Warning:**
>
> Do not 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 XenServer host to "forget" is dead; however, if the XenServer host is live and part of the pool, you should use **xe pool-eject** instead.

host-get-system-status

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

Download system status information into the specified file. The optional parameter `entries` is a comma-separated list of system status entries, taken from the capabilities XML fragment returned by the **host-get-system-status-capabilities** command. See the section called “host-get-system-status-capabilities” for details. If not specified, all system status information is saved in the file. The parameter `output` may be *tar.bz2* (the default) or *zip*; if this parameter is not specified, the file is saved in `tar.bz2` form.

The host(s) on which this operation should be performed are selected 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="xenserver-logs"  \
           max-size="150425200" max-time="-1" min-size="150425200" min-time="-1" \
     pii="maybe"/>
   <capability content-type="text/plain" default-checked="yes" \
     key="xenserver-install" max-size="51200" max-time="-1" min-size="10240" \
     min-time="-1" pii="maybe"/>
   ...
 </system-status-capabilities>
```

Each capability entity has a number of attributes.

Attribute	Description
key	A unique identifier for the capability.
content-type	Can be either *text/plain* or *application/data*. Indicates whether a UI can render the entries for human consumption.
default-checked	Can be either *yes* or *no*. Indicates whether a UI should select this entry by default.
min-size, max-size	Indicates an approximate range for the size, in bytes, of this entry. *-1* indicates that the size is unimportant.
min-time, max-time	Indicate an approximate range for the time, in seconds, taken to collect this entry. `-1` indicates the time is unimportant.
pii	Personally identifiable information. Indicates whether the entry would have information that would identify the system owner, or details of their network topology. This is one of: • *no*: no PII will be in these entries • *yes*: PII will likely or certainly be in these entries • *maybe*: you might wish to audit these entries for PII • *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. 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-apply-edition

`host-apply-edition` [host-uuid=*<XenServer_host_UUID>*] [edition=xenserver_edition=*<"free"><"advanced"><"enterprise"><"platinum"><"enterprise-xd">*]

Assigns a XenServer license to a host server. When you assign a license, XenServer contacts the Citrix License Server and requests the specified type of license. If a license is available, it is then checked out from the license server.

For Citrix XenServer for XenDesktop editions, use *<"enterprise-xd">*.

For initial licensing configuration, see also **license-server-address** and **license-server-port**.

host-license-add

`host-license-add` [license-file=*<path/license_filename>*] [host-uuid=*<XenServer_host_UUID>*]

For XenServer (free edition), use to parses a local license file and adds it to the specified XenServer host.

host-license-view

`host-license-view` [host-uuid=*<XenServer_host_UUID>*]

Displays the contents of the XenServer host license.

host-logs-download

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

Download a copy of the logs of the specified XenServer hosts. The copy is saved by default in a time-stamped 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 parameters listed at the beginning of this section.

> 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 (such as the XenCenter). Operates directly on the XenServer host the CLI is connected to, and is not forwarded to the pool master if applied to a member XenServer host.

Warning:

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

host-management-reconfigure

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

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

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

If the UUID of a PIF object is specified, the XenServer host 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 XenServer host.

host-power-on

`host-power-on` [host=*<host_uuid>*]

Turns on power on XenServer hosts with Host Power On functionality enabled. Before using this command, **host-set-power-on** must be enabled on the host.

host-get-cpu-features

`host-get-cpu-features` {features=*<pool_master_cpu_features>*} [uuid=*<host_uuid>*]

Prints a hexadecimal representation of the host's physical-CPU features.

host-set-cpu-features

`host-set-cpu-features` {features=*<pool_master_cpu_features>*} [uuid=*<host_uuid>*]

Attempts to mask the host's physical-CPU features to match the given features. The given string must be a 32-digit hexadecimal number (optionally containing spaces), as given by the `host-get-cpu-features` command.

host-set-power-on

`host-set-power-on` {host=*<host uuid>* {power-on-mode=*<"">* *<"wake-on-lan">* *<"iLO">* *<"DRAC">* *<"custom">* } | [power-on-config=*<"power_on_ip"><"power_on_user"><"power_on_password_secret">*] }

Use to enable Host Power On functionality on XenServer hosts that are compatible with remote power solutions. Workload Balancing requires Host Power On functionality is enabled for it to turn off underused hosts in Maximum Density mode. When using the **host-set-power-on** command, you must specify the type of power management solution on the host (that is, the *<power-on-mode>*). Then specify configuration options using the *<power-on-config>* argument and its associated key-value pairs. To use the secrets feature to store your password, specify the key "power_on_password_secret".

host-reboot

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

Reboot the specified XenServer hosts. The specified 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 parameters listed at the beginning of this section.

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

host-restore

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

Restore a backup named *`file-name`* of the XenServer host control software. Note that the use of the word "restore" here does not mean a full restore in the usual sense, it merely means that the compressed backup file has been uncompressed and unpacked onto the secondary partition. After you've done a **xe host-restore**, you have to boot the Install CD and use its **Restore 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 parameters listed at the beginning of this section.

host-set-hostname-live

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

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

host-shutdown

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

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

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

If the specified XenServer hosts are members of a pool, the loss of connectivity on shutdown will be handled and the pool will recover when the XenServer hosts returns. If you shut down a pool member, other members and the master will continue to function. If you shut down the master, the pool will be out of action until the master is rebooted and back on line, at which point the members will reconnect and synchronize with the master, or until one of the members is made into the master. If HA is enabled for the pool, one of the members will be made into a master automatically. If HA is disabled, you must manually 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 XenServer 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 parameters listed at the beginning of this section.

host-data-source-list

`host-data-source-list` [*<host-selectors>*=*<host selector value>*...]

List the data sources that can be recorded for a host.

Select the host(s) on which to perform this operation by using the standard selection mechanism (see host selectors). Optional arguments can be any number of the host parameters listed at the beginning of this section. If no parameters to select hosts are given, the operation will be performed on all hosts.

Data sources have two parameters — `standard` and `enabled` — which can be seen by the output of this command. If a data source has `enabled` set to `true`, then the metrics are currently being recorded to the performance database. If a data source has `standard` set to `true`, then the metrics are recorded to the performance database *by default* (and so, `enabled` will also be set to `true` for this data source). If a data source has `standard` set to `false`, then the metrics are *not* recorded to the performance database by default (and so, `enabled` will also be set to `false` for this data source).

To start recording data source metrics to the performance database, run the **host-data-source-record** command. This will set `enabled` to `true`. To stop, run the **host-data-source-forget**. This will set `enabled` to `false`.

host-data-source-record

`host-data-source-record` data-source=*<name_description_of_data-source>* [*<host-selectors>*=*<host selector value>*...]

Record the specified data source for a host.

This operation writes the information from the data source to the persistent performance metrics database of the specified host(s). For performance reasons, this database is distinct from the normal agent database.

Select the host(s) on which to perform this operation by using the standard selection mechanism (see host selectors). Optional arguments can be any number of the host parameters listed at the beginning of this section. If no parameters to select hosts are given, the operation will be performed on all hosts.

host-data-source-forget

`host-data-source-forget` data-source=*<name_description_of_data-source>* [*<host-selectors>*=*<host selector value>*...]

Stop recording the specified data source for a host and forget all of the recorded data.

Select the host(s) on which to perform this operation by using the standard selection mechanism (see host selectors). Optional arguments can be any number of the host parameters listed at the beginning of this section. If no parameters to select hosts are given, the operation will be performed on all hosts.

host-data-source-query

`host-data-source-query` data-source=*<name_description_of_data-source>* [*<host-selectors>*=*<host selector value>*...]

Display the specified data source for a host.

Select the host(s) on which to perform this operation by using the standard selection mechanism (see host selectors). Optional arguments can be any number of the host parameters listed at the beginning of this section. If no parameters to select hosts are given, the operation will be performed on all hosts.

Log Commands

Commands for working with logs.

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. Messages are created to notify users of significant events, and are displayed in XenCenter as system alerts.

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-destroy

`message-destroy` {uuid=*<message_uuid>*}

Destroys an existing message. You can build a script to destroy all messages. For example:

```
# Dismiss all alerts   \
    IFS=","; for m in $(xe message-list params=uuid --minimal); do  \
    xe message-destroy uuid=$m  \
    done
```

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 Parameter 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 XenServer hosts to this network	read only set parameter
bridge	name of the bridge corresponding to this network on the local XenServer host	read only
other-config:static-routes	comma-separated list of *<subnet>*/*<netmask>*/*<gateway>* formatted entries specifying the gateway address through which to route subnets. For example, setting `other-config:static-routes` to `172.16.0.0/15/192.168.0.3,172.1` 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`.	read write

Parameter Name	Description	Type
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 XenServer host patches (updates). These are for the standard non-OEM editions of XenServer for commands relating to updating the OEM edition of XenServer, see the section called “Update Commands” for details.

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

Patch Parameters

Patches have the following parameters:

Parameter Name	Description	Type
uuid	The unique identifier/object reference for the patch	read only
host-uuid	The unique identifier for the XenServer host to query	read only

Parameter Name	Description	Type
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 XenServer host.

patch-pool-apply

`patch-pool-apply` uuid=*<patch_uuid>*

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

patch-precheck

`patch-precheck` uuid=*<patch_uuid>* host-uuid=*<host_uuid>*

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

patch-upload

`patch-upload` file-name=*<patch_filename>*

Upload a specified patch file to the XenServer host. This prepares a patch to be applied. On success, the UUID of the uploaded patch is printed out. If the patch has previously been uploaded, a PATCH_ALREADY_EXISTS error is returned instead and the patch is not uploaded again.

PBD Commands

Commands for working with PBDs (Physical Block Devices). These are the software objects through which the XenServer host accesses storage repositories (SRs).

The PBD objects can be listed with the standard object listing command (**xe pbd-list**), and the parameters manipulated with the standard parameter commands. See the section called “Low-level Parameter 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 XenServer 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 *Storage*.

pbd-destroy

`pbd-destroy` uuid=*<uuid_of_pbd>*

Destroy the specified PBD.

pbd-plug

`pbd-plug` uuid=*<uuid_of_pbd>*

Attempts to plug in the PBD to the XenServer host. If this succeeds, the referenced SR (and the VDIs contained within) should then become visible to the XenServer host.

pbd-unplug

`pbd-unplug` uuid=*<uuid_of_pbd>*

Attempt to unplug the PBD from the XenServer host.

PIF Commands

Commands for working with PIFs (objects representing the physical network interfaces).

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

PIF Parameters

PIFs have the following parameters:

Parameter Name	Description	Type
uuid	the unique identifier/object reference for the PIF	read only
device	machine-readable name of the interface (for example, eth0)	read only
MAC	the MAC address of the PIF	read only
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 XenServer host to which this PIF is connected	read only
host-name-label	the name of the XenServer host to which this PIF is connected	read only
IP-configuration-mode	type of network address configuration used; DHCP or static	read only
IP	IP address of the PIF, defined here if IP-configuration-mode is static; undefined if DHCP	read only

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
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

Parameter Name	Description	Type
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 XenServer host>* mac=*<mac_address_for_pif>* device=*<machine-readable name of the interface (for example, eth0)>*

Create a new PIF object representing a physical interface on the specified XenServer host.

pif-plug

`pif-plug` uuid=*<uuid_of_pif>*

Attempt to bring up the specified physical interface.

pif-reconfigure-ip

`pif-reconfigure-ip` uuid=*<uuid_of_pif>* [mode=*<dhcp>* | mode=*<static>*]
gateway=*<network_gateway_address>* IP=*<static_ip_for_this_pif>*
netmask=*<netmask_for_this_pif>* [DNS=*<dns_address>*]

Modify the IP address of the PIF. For static IP configuration, set the *`mode`* parameter to `static`, with the *`gateway`*, *`IP`*, and *`netmask`* parameters set to the appropriate values. To use DHCP, set the *`mode`* parameter to *DHCP* and leave the static parameters undefined.

Note:

Using static IP addresses on physical network interfaces connected to a port on a switch using Spanning Tree Protocol with STP Fast Link turned off (or unsupported) results in a period during which there is no traffic.

pif-scan

`pif-scan` host-uuid=*<UUID of XenServer host>*

Scan for new physical interfaces on a XenServer host.

pif-unplug

`pif-unplug` uuid=*<uuid_of_pif>*

Attempt to bring down the specified physical interface.

Pool Commands

Commands for working with pools. A *pool* is an aggregate of one or more XenServer hosts. A pool uses one or more shared storage repositories so that the VMs running on one XenServer host in the pool can be migrated in near-real time (while still running, without needing to be shut down and brought back up) to another XenServer host in the pool. Each XenServer host is really a pool consisting of a single member by default. When a XenServer host is joined to a pool, it is designated as a member, and the pool it has joined becomes the master for the pool.

The singleton pool object can be listed with the standard object listing command (**xe pool-list**), and its parameters manipulated with the standard parameter commands. See the section called “Low-level Parameter Commands” for details.

Pool Parameters

Pools have the following parameters:

Parameter Name	Description	Type
uuid	the unique identifier/object reference for the pool	read only
name-label	the name of the pool	read/write
name-description	the description string of the pool	read/write
master	the unique identifier/object reference of XenServer host designated as the pool's master	read only
default-SR	the unique identifier/object reference of the default SR for the pool	read/write
crash-dump-SR	the unique identifier/object reference of the SR where any crash dumps for pool members are saved	read/write
metadata-vdis	all known metadata VDIs for the pool	read only
suspend-image-SR	the unique identifier/object reference of the SR where suspended VMs on pool members are saved	read/write

Parameter Name	Description	Type
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 XenServer host to become new master>*

Instruct the specified member XenServer host to become the master of an existing pool. This performs an orderly hand over of the role of master host to another host in the resource pool. This command only works when the current master is online, and is not a replacement for the emergency mode commands listed below.

pool-dump-database

`pool-dump-database` file-name=*<filename_to_dump_database_into_(on_client)>*

Download a copy of the entire pool database and dump it into a file on the client.

pool-eject

```
pool-eject host-uuid=<UUID of XenServer host to eject>
```

Instruct the specified XenServer host to leave an existing pool.

pool-emergency-reset-master

```
pool-emergency-reset-master master-address=<address of the pool's master XenServer host>
```

Instruct a slave member XenServer host to reset its master address to the new value and attempt to connect to it. This command should not be run on master hosts.

pool-emergency-transition-to-master

```
pool-emergency-transition-to-master
```

Instruct a member XenServer host to become the pool master. This command is only accepted by the XenServer host if it has transitioned to emergency mode, meaning it is a member of a pool whose master has disappeared from the network and could not be contacted for some number of retries.

Note that this command may cause the password of the host to reset if it has been modified since joining the pool (see the section called “User Commands”).

pool-ha-enable

```
pool-ha-enable heartbeat-sr-uuids=<SR_UUID_of_the_Heartbeat_SR>
```

Enable High Availability on the resource pool, using the specified SR UUID as the central storage heartbeat repository.

pool-ha-disable

```
pool-ha-disable
```

Disables the High Availability functionality on the resource pool.

pool-join

```
pool-join master-address=<address> master-username=<username> master-password=<password>
```

Instruct a XenServer host to join an existing pool.

pool-recover-slaves

```
pool-recover-slaves
```

Instruct the pool master to try and reset the master address of all members currently running in emergency mode. This is typically used after **pool-emergency-transition-to-master** has been used to set one of the members as the new master.

pool-restore-database

```
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 Parameter Commands" for details.

SM Parameters

SMs have the following parameters:

Parameter Name	Description	Type
uuid	the unique identifier/object reference for the SM plugin	read only
name-label	the name of the SM plugin	read only
name-description	the description string of the SM plugin	read only
type	the SR type that this plugin connects to	read only
vendor	name of the vendor who created this plugin	read only
copyright	copyright statement for this SM plugin	read only
required-api-version	minimum SM API version required on the XenServer host	read only
configuration	names and descriptions of device configuration keys	read only
capabilities	capabilities of the SM plugin	read only
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 Parameter 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-Utilization	physical space currently utilized on this SR, in bytes. Note that for sparse disk formats, physical utilization may be less than virtual allocation	read only
physical-size	total physical size of the SR, in bytes	read only
type	type of the SR, used to specify the SR backend driver to use	read only
introduced-by	the drtask (if any) which introduced the SR	read onlyk
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, Citrix 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 XenServer 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

Parameter Name	Description	Type
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=*<XenServer 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 XenServer host. If *`shared`* is set to `true`, a PBD is created for each XenServer host in the pool; if *`shared`* is not specified or set to `false`, a PBD is created only for the XenServer host specified with *`host-uuid`*.

The exact *`device-config`* parameters differ depending on the device *`type`*. See *Storage* for details of these parameters across the different storage backends.

sr-destroy

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

Destroys the specified SR on the XenServer host.

sr-enable-database-replication

`sr-enable-database-replication` uuid=*<sr_uuid>*

Enables xapi database replication to the specified (shared) SR. For example:

```
xe sr-enable-database-replication uuid=<sr-uuid>
```

sr-disable-database-replication

`sr-disable-database-replication` uuid=*<sr_uuid>*

Disables xapi database replication to the specified SR. For example:

```
xe sr-enable-database-replication uuid=<sr-uuid>
```

sr-forget

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

The `xapi` agent forgets about a specified SR on the XenServer host, meaning that the SR is detached and you cannot access VDIs on it, but it remains intact on the source media (the data is not lost).

sr-introduce

`sr-introduce` name-label=*<name>*
physical-size=*<physical_size>*

type=*<type>*
content-type=*<content_type>*
uuid=*<sr_uuid>*

Just places an SR record into the database. The `device-config` parameters are specified by `device-config:`*`<parameter_key>`*`=`*`<parameter_value>`*, for example:

```
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 *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 Parameter Commands” for details.

Task Parameters

Tasks have the following parameters:

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

Parameter Name	Description	Type
name-description	the description string of the Task	read only
resident-on	the unique identifier/object reference of the host on which the task is running	read only
status	current status of the Task	read only
progress	if the Task is still pending, this field contains the estimated percentage complete, from 0. to 1. If the Task has completed, successfully or unsuccessfully, this should be 1.	read only
type	if the Task has 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
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. XenServer ships with a base set of templates, which range from generic "raw" VMs that can boot an OS vendor installation CD (RHEL, CentOS, SLES, Windows) to complete pre-configured OS instances (the "Demo Linux VM" template). With XenServer you can create VMs, configure them in standard forms for your particular needs, and save a copy of them as templates for future use in VM deployment.

The template objects can be listed with the standard object listing command (**xe template-list**), and the parameters manipulated with the standard parameter commands. See the section called “Low-level Parameter 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	read/write
is-control-domain	true if this is a control domain (domain 0 or a driver domain)	read only
power-state	current power state; always `halted` for a template	read only
power-state	current power state; always `halted` for a template	read only
memory-dynamic-max	dynamic maximum memory in bytes. Currently unused, but if changed the following constraint must be obeyed: *memory_static_max* >= *memory_dynamic_max* >= *memory_dynamic_min* >= *memory_static_min*.	read/write
memory-dynamic-min	dynamic minimum memory in bytes. Currently unused, but if changed the same constraints for *memory-dynamic-max* must be obeyed.	read/write
memory-static-max	statically-set (absolute) maximum memory in bytes. This is the main value used to determine the amount of memory assigned to a VM.	read/write

Parameter Name	Description	Type
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

Parameter Name	Description	Type

VCPUs-params	configuration parameters for the selected VCPU policy.
	You can tune a VCPU's pinning with
	`xe vm-param-set \`
	`uuid=<vm_uuid> \`
	`VCPUs-params:mask=1,2,3`
	A VM created from this template will then run on physical CPUs 1, 2, and 3 only.
	You can also tune the VCPU priority (xen scheduling) with the *cap* and *weight* parameters; for example
	`xe vm-param-set \`
	`uuid=<vm_uuid> \`
	`VCPUs-params:weight=512`
	`xe vm-param-set \`
	`uuid=<vm_uuid> \`
	`VCPUs-params:cap=100`
	A VM based on this template with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended XenServer host. Legal weights range from 1 to 65535 and the default is 256.
	The cap optionally fixes the maximum amount of CPU a VM based on this template will be able to consume, even if the XenServer host has idle CPU cycles. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc. The default, 0, means there is no upper cap.
		read/write map parameter

VCPUs-max	maximum number of VCPUs	read/write
VCPUs-at-startup	boot number of VCPUs	read/write
actions-after-crash	action to take if a VM based on this template crashes	read/write
console-uuids	virtual console devices	read only set parameter
platform	platform-specific configuration	read/write map parameter

Parameter Name	Description	Type
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
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

Parameter Name	Description	Type
resident-on	the XenServer host on which a VM based on this template is currently resident; appears as `<not in database>` for a template	read only
affinity	a XenServer host which a VM based on this template has preference for running on; used by the **xe vm-start** command to decide where to run the VM	read/write
other-config	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 z`, where `z` is the single-letter military timezone indicator, for example, Z for UTC (GMT); set to 1 Jan 1970 Z (beginning of Unix/ POSIX epoch) for a template	read only
install-time	timestamp of the date and time that the metrics for a VM based on this template were read, in the form `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-Utilization	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

Parameter Name	Description	Type
PV-drivers-version	the versions of the paravirtualized drivers for a VM based on this template; appears as <not in database> for a template	read only map parameter
PV-drivers-up-to-date	flag for latest version of the paravirtualized drivers for a VM based on this template; appears as <not in database> for a template	read only
memory	memory metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
disks	disk metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
networks	network metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
other	other metrics reported by the agent on a VM based on this template; appears as <not in database> for a template	read only map parameter
guest-metrics-last-updated	timestamp when the last write to these fields was performed by the in-guest agent, in the form yyyymmddThh:mm:ss z, where z is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
actions-after-shutdown	action to take after the VM has shutdown	read/write
actions-after-reboot	action to take after the VM has rebooted	read/write
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

Parameter Name	Description	Type
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
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.

Update Commands

Commands for working with updates to the OEM edition of XenServer. For commands relating to updating the standard non-OEM editions of XenServer, see the section called “Patch (Update) Commands” for details.

update-upload

`update-upload` file-name=*<name_of_upload_file>*

Streams a new software image to a OEM edition XenServer host. You must then restart the host for this to take effect.

User Commands

user-password-change

`user-password-change` old=*<old_password>* new=*<new_password>*

Changes the 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 Parameter 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

Parameter Name	Description	Type
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	error/success code associated with the last attach operation	read only
status-detail	error/success information associated with the last attach operation status	read only
qos_algorithm_type	the QoS algorithm to use	read/write
qos_algorithm_params	parameters for the chosen QoS algorithm	read/write map parameter
qos_supported_algorithms	supported QoS algorithms for this VBD	read only set parameter
io_read_kbs	average read rate in kB 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 Parameter 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-Utilization	amount of physical space that the VDI is currently taking up on the SR, in bytes	read only
type	type of VDI, for example, System or User	read only
sharable	true if this VDI may be shared	read only
read-only	true if this VDI can only be mounted read-only	read only
storage-lock	true if this VDI is locked at the storage level	read only
parent	references the parent VDI, if this VDI is part of a chain	read only
missing	true if SR scan operation reported this VDI as not present	read only
other-config	additional configuration information for this VDI	read/write map parameter
sr-name-label	name of the containing storage repository	read only
location	location information	read only
managed	true if the VDI is managed	read only

Parameter Name	Description	Type
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
metadata-of-pool	the uuid of the pool which created this metadata VDI	read only
metadata-latest	flag indicating whether the VDI contains the latest known metadata for this pool	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 backend. In normal operation, you should be using **vdi-destroy** instead.

vdi-import

`vdi-import` uuid=*<uuid_of_vdi>* filename=*<filename_of_raw_vdi>*

Import a raw VDI.

vdi-introduce

`vdi-introduce` uuid=*<uuid_of_vdi>*
sr-uuid=*<uuid_of_sr_to_import_into>*
name-label=*<name_of_the_new_vdi>*
type=*<system | user | suspend | crashdump>*
location=*<device_location_(varies_by_storage_type)>*
[name-description=*<description_of_vdi>*]
[sharable=*<yes | no>*]
[read-only=*<yes | no>*]
[other-config=*<map_to_store_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 template creation 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 Parameter 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

Parameter Name	Description	Type

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:
	`xe vif-param-set \`
	`uuid=<vif_uuid> \`
	`other-config:mtu=9000`
		read only

currently-attached	true if the device is currently attached	read only
qos_algorithm_type	QoS algorithm to use	read/write
qos_algorithm_params	parameters for the chosen QoS algorithm	read/write map parameter
qos_supported_algorithms	supported QoS algorithms for this VIF	read only set parameter
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

Parameter Name	Description	Type
other-config:promiscuous	`true` to a VIF to be promiscuous on the bridge, so that it sees all traffic over the bridge. Useful for running an Intrusion Detection System (IDS) or similar in a VM.	read write
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 XenServer 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>*. An easy way to get the uuid of an actual VM is to, for example, execute **xe vm-list power-state=running**. (The full list of fields that can be matched can be obtained by the command **xe vm-list params-all**.) 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*.

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 Parameter Commands" for details.

VM Parameters

VMs have the following parameters:

> **Note:**
>
> All writable 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
appliance	the appliance/vApp to which the VM belongs	read/write
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
order	start order for vApp startup/ shutdown and for startup after HA failover	read/write

Parameter Name	Description	Type
version	the number of times this VM has been recovered - if a user wishes to over write a new VM with an older version, then they must call vm-recover	read only
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	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
start delay	the delay to wait before a call to start up the VM returns	read/write
shutdown-delay	the delay to wait before a call to shutdown the VM returns	read/write
memory-dynamic-max	dynamic maximum in bytes	read/write
memory-dynamic-min	dynamic minimum in bytes	read/write
memory-static-max	statically-set (absolute) maximum in bytes. If you want to change this value, the VM must be shut down.	read/write
memory-static-min	statically-set (absolute) minimum in bytes. If you want to change this value, the VM must be shut down.	read/write
suspend-VDI-uuid	the VDI that a suspend image is stored on	read only

Parameter Name	Description	Type

VCPUs-params	configuration parameters for the selected VCPU policy.
	You can tune a VCPU's pinning with
	`xe vm-param-set \`
	`uuid=<vm_uuid> \`
	`VCPUs-params:mask=1,2,3`
	The selected VM will then run on physical CPUs 1, 2, and 3 only.
	You can also tune the VCPU priority (xen scheduling) with the *cap* and *weight* parameters; for example
	`xe vm-param-set \`
	`uuid=<template_uuid> \`
	`VCPUs-params:weight=512`
	`xe vm-param-set \`
	`uuid=<template UUID> \`
	`VCPUs-params:cap=100`
	A VM with a weight of 512 will get twice as much CPU as a domain with a weight of 256 on a contended XenServer host. Legal weights range from 1 to 65535 and the default is 256.
	The cap optionally fixes the maximum amount of CPU a VM will be able to consume, even if the XenServer host has idle CPU cycles. The cap is expressed in percentage of one physical CPU: 100 is 1 physical CPU, 50 is half a CPU, 400 is 4 CPUs, etc. The default, 0, means there is no upper cap.
		read/write map parameter

VCPUs-max	maximum number of virtual CPUs.	read/write
VCPUs-at-startup	boot number of virtual CPUs	read/write

Parameter Name	Description	Type
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 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
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

Parameter Name	Description	Type
PV-kernel	path to the kernel	read/write
PV-ramdisk	path to the initrd	read/write
PV-args	string of kernel command line arguments	read/write
PV-legacy-args	string of arguments to make legacy VMs boot	read/write
PV-bootloader	name of or path to bootloader	read/write
PV-bootloader-args	string of miscellaneous arguments for the bootloader	read/write
last-boot-CPU-flags	describes the CPU flags on which the VM was last booted	read only
resident-on	the XenServer host on which a VM is currently resident	read only
affinity	a XenServer host which the VM has preference for running on; used by the **xe vm-start** command to decide where to run the VM	read/write
other-config	A list of key/value pairs that specify additional configuration parameters for the VM For example, a VM will be started automatically after host boot if the other-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 timezone indicator, for example, Z for UTC (GMT)	read only
install-time	timestamp of the date and time that the metrics for the VM were read, in the form `yyyymmddThh:mm:ss z`, where `z` is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only
memory-actual	the actual memory being used by a VM	read only

Parameter Name	Description	Type
VCPUs-number	the number of virtual CPUs assigned to the VM For a paravirtualized Linux VM, this number can differ from *`VCPUS-max`* and can be changed without rebooting the VM 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. Note that performance will drop sharply if you set *`VCPUs-number`* to a value greater than the number of physical CPUs on the XenServer host.	read only
VCPUs-Utilization	a list of virtual CPUs and their weight	read only map parameter
os-version	the version of the operating system for the VM	read only map parameter
PV-drivers-version	the versions of the paravirtualized drivers for the VM	read only map parameter
PV-drivers-up-to-date	flag for latest version of the paravirtualized drivers for the VM	read only
memory	memory metrics reported by the agent on the VM	read only map parameter
disks	disk metrics reported by the agent on the VM	read only map parameter
networks	network metrics reported by the agent on the VM	read only map parameter
other	other metrics reported by the agent on the VM	read only map parameter
guest-metrics-last-updated	timestamp when the last write to these fields was performed by the in-guest agent, in the form `yyyymmddThh:mm:ssz`, where `z` is the single-letter military timezone indicator, for example, Z for UTC (GMT)	read only

Parameter Name	Description	Type
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
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-assert-can-be-recovered

`vm-assert-can-be-recovered` *<uuid>* [*<database>*] *<vdi-uuid>*

Tests whether storage is available to recover this VM.

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-list

`vm-data-source-list` [*<vm-selector>*=*<vm selector value>*...]

List the data sources that can be recorded for a VM.

Select the VM(s) on which to perform this operation by 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 no parameters to select hosts are given, the operation will be performed on all VMs.

Data sources have two parameters — `standard` and `enabled` — which can be seen by the output of this command. If a data source has `enabled` set to `true`, then the metrics are currently being recorded to the performance database. If a data source has `standard` set to `true`, then the metrics are recorded to the performance database *by default* (and so, `enabled` will also be set to `true` for this data source). If a data source has `standard` set to `false`, then the metrics are *not* recorded to the performance database by default (and so, `enabled` will also be set to `false` for this data source).

To start recording data source metrics to the performance database, run the **vm-data-source-record** command. This will set `enabled` to `true`. To stop, run the **vm-data-source-forget**. This will set `enabled` to `false`.

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 operation writes the information from the data source to the persistent performance metrics database of the specified VM(s). For performance reasons, this database is distinct from the normal agent database.

Select the VM(s) on which to perform this operation by 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 no parameters to select hosts are given, the operation will be performed on all VMs.

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.

Select the VM(s) on which to perform this operation by 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 no parameters to select hosts are given, the operation will be performed on all VMs.

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.

Select the VM(s) on which to perform this operation by 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 no parameters to select hosts are given, the operation will be performed on all VMs.

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 XenServer 3.2 and is used by some third-party vendors to provide virtual appliances. This format uses a directory to store the VM data, so set *filename* to the root directory of the XVA export and not an actual file. Subsequent exports of the imported legacy guest will automatically be upgraded to the new filename-based format, which stores much more data about the configuration of the VM.

> **Note:**
>
> The older directory-based XVA format does not fully preserve all the VM attributes. In particular, imported VMs will not have any virtual network interfaces attached by default. If networking is required, create one using **vif-create** and **vif-plug**.

If the *metadata* is `true`, then a previously exported set of metadata can be imported without their associated disk blocks. Metadata-only import will fail if any VDIs cannot be found (named by SR and *VDI.location*) unless the *--force* option is specified, in which case the import will proceed regardless. If disks can be mirrored 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> ]
[ copy-bios-strings-from=<uuid of host> ]
```

Install or clone a VM from a template. Specify the template name using either the *template-uuid* or *template* argument. Specify an SR using either the *sr-uuid* or *sr-name-label* argument. Specify to install BIOS-locked media using the *copy-bios-strings-from* argument.

> **Note:**
>
> When installing from a template that has existing disks, by default, new disks will be created in the same SR as these existing disks. Where the SR supports it, these will be fast copies. If a different SR is specified on the command line, the new disks will be created there. In this case a fast copy is not possible and the disks will be full copies.
>
> When installing from a template that does not have existing disks, any new disks will be created in the SR specified, or the pool default SR if not specified.

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 Citrix 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 XenServer host memory will decrease according to the multiplier, and the *`HVM_shadow_multiplier`* field will be updated with the actual value which Xen has assigned to the VM. If there is not enough XenServer host memory free, then an error will be returned.

The VMs on which this operation should be performed are selected using the standard selection mechanism (see VM selectors for more information).

vm-migrate

`vm-migrate` [[host-uuid=*<destination XenServer host UUID>*] | [host=*<name or UUID of destination XenServer host>*]] [*<vm-selector>*=*<vm_selector_value>*...] [live=*<true | false>*]

Migrate the specified VMs between physical hosts. The *`host`* parameter can be either the name or the UUID of the XenServer host.

By default, the VM will be suspended, migrated, and resumed on the other host. The *`live`* parameter activates XenMotion and keeps the VM running while performing the migration, thus minimizing VM downtime to less than a second. In some circumstances such as extremely memory-heavy workloads in the VM, XenMotion automatically falls back into the default mode and suspends the VM for a brief period of time before completing the memory transfer.

The VM or VMs on which this operation should be performed are selected 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-recover

`vm-recover` *<vm-uuid>* [*<database>*] [*<vdi-uuid>*] [*<force>*]

Recovers a VM from the database contained in the supplied VDI.

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=*<XenServer 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=*<XenServer 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 XE 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> \
xenserver_username=<pool_master_username> \
xenserver_password=<pool_master_password>
```

Initializes the connection between the Workload Balancing server and the XenServer pool.

> **Note:**
>
> Initializing a pool requires running two commands. After executing the **pool-initialize-wlb** command, execute **xe pool-param-set wlb-enabled=true uuid=<pool-uuid>**[].

pool-param-set other-config

Use the **pool-param-set other-config** command to specify the timeout when communicating with the Workload Balancing 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>
```

pool-retrieve-wlb-diagnostics

Use the **pool-retrieve-wlb-diagnostics** command to return the full contents of the Workload Balancing Log file for the Workload Balancing server that the pool is currently using. This command also returns some version information.

```
xe pool-retrieve-wlb-diagnostics \
```

host-retrieve-wlb-evacuate-recommendations

```
host-retrieve-wlb-evacuate-recommendations uuid=<host_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-certificate-list

XenServer Workload Balancing component lets you use certificates to secure communication between XenServer pools and the Workload Balancing server. You can either use the default test certificate, which is created automatically during Workload Balancing configuration, or you can specify a certificate from a Trusted Authority.

To use a Trusted Authority certificate, the certificate must be in X.509 format and comply with the guidelines provided earlier in the *Workload Balancing Administrator's Guide*. If you want to verify the presence of a certificate from a Trusted Authority, you must:

```
pool-certificate-list
```

Lists all installed SSL certificates.

pool-certificate-install

```
pool-certificate-install filename=<certificatefilename>
```

Run this command on the pool to install the certificate on the Workload Balancing virtual appliance on the pool master.

pool-certificate-sync

```
pool-certificate-sync
```

Run this command on the pool, after running the pool-certificate-install command, to make sure the certificate and certificate revocation lists are synchronized from the pool master to all slaves on the pool.

pool-param-set

```
pool-param-set wlb-verify-cert=<true> uuid=<uuid_of_pool>
```

Run this command on the pool, after running the pool-certificate-sync command, to make XenServer always verify the certificate when communicating with the Workload Balancing server.

> **Tip:**
>
> Pressing the **Tab** key automatically populates the UUID of the pool.

pool-deconfigure-wlb

Permanently deletes all workload balancing configuration.

pool-retrieve-wlb-configuration

Prints all Workload Balancing configuration information to standard out.

pool-retrieve-wlb-recommendations

Prints all Workload Balancing recommendations to standard out.

pool-retrieve-wlb-report

Gets a Workload Balancing report of the specified type and saves it to the specified file. XenCenter uses this command when executing Workload Balancing reports. However, running this command from the CLI does not output report data in a graphical format. Outputting reports requires using the XenCenter **Pool** > **View Workload Reports** menu command. The available reports are:

- `pool_health`
- `host_health_history`
- `optimization_performance_history`
- `pool_optimization_history`
- `pool_health_history`
- `vm_chargeback_history`
- `pool_audit_history`
- `vm_movement_history`
- `vm_performance_history`

Example usage for each report type is shown below. The *`utcoffset`* parameter specifies the number of hours the pool's time zone is ahead or behind Coordinated Universal Time (UTC). For example, *<-5>* represents the Eastern Standard Time zone in North America, which is five hours behind UTC. The `start` parameter and *end* parameters specify the number of hours to report. For example specifying *`start=-3`* and *`end=0`* will cause Workload Balancing to report on the last three hours of activity. *`LocaleCode`* represents the language for the report output. You can enter either *<en>* for English or *<ja>* for Japanese.

```
xe pool-retrieve-wlb-report report=pool_health \
LocaleCode=<en> \
Start=<-3> \
End=<0> \
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c> \
UTCOffset=<-5> \
filename=/<pool_health>.txt \
```

```
xe pool-retrieve-wlb-report report=host_health_history \
Start=<-3> \
End=<0> \
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c> \
HostID=<471626e5-8e8b-429b-9990-6013726d7e86> \
UTCOffset=<-5> \
filename=/<host_health_history.txt> \
```

```
xe pool-retrieve-wlb-report report=optimization_performance_history \
LocaleCode=<en> \
Start=-3 \
End=0 \
PoolID=c75f9ee9-422f-9cde-4781-24f6cbd9279c \
UTCOffset=-5 \
filename=</optimization_performance_history>.txt \
xe pool-retrieve-wlb-report report=pool_health_history \
LocaleCode=<en> \
Start=<-3> \
End=<0> \
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c> \
UTCOffset=<-5> \
filename=</pool_health_history>.txt \
```

```
xe pool-retrieve-wlb-report report=vm_movement_history \
LocaleCode=<en> \
Start=<-3> \
End=<0> \
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c> \
UTCOffset=<-5> \
filename=</vm_movement_history>.txt \
```

```
xe pool-retrieve-wlb-report report=vm_performance_history \
LocaleCode=<en> \
Start=<-3> \
End=<0> \
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c> \
HostID=<471626e5-8e8b-429b-9990-6013726d7e86> \
UTCOffset=<-5> \
filename=</vm_performance_history>.txt \
```

```
xe pool-retrieve-wlb-report report=pool_audit_history \
LocaleCode=<en> \
Start=<-3> \
End=<0> \
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c> \
UTCOffset=<-5> \
filename=</pool_audit_history>.txt \
```

```
xe pool-retrieve-wlb-report report=pool_optimization_history
LocaleCode=<en>
Start=<-3>
End=<0>
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c>
UTCOffset=<-5>
filename=</pool_optimization_history>.txt
```

```
xe pool-retrieve-wlb-report report=vm_chargeback_history
LocaleCode=<en>
Start=<-3>
End=<0>
PoolID=<c75f9ee9-422f-9cde-4781-24f6cbd9279c>
UTCOffset=<-5>
filename=</vm_chargeback_history>.txt
```

pool-send-wlb-configuration

Modifies Workload Balancing configuration settings, including thresholds, Workload Balancing power-management settings, and weightings. It is not mandatory to configure all settings with the command. You can configure only some parameters, but not all, if desired.

Before using the pool-send-wlb-configuration command, learn about the default values on your system by running pool-retrieve-wlb-configuration.

When you run the pool-retrieve-wlb-configuration command, additional parameters appear that are not documented in this section. Citrix does not recommend editing these parameters.

```
pool-send-wlb-configuration
[config:HostMemoryThresholdCritical=<HostCpuThresholdCritical=value>\
config:HostMemoryThresholdHigh=<HostMemoryThresholdHigh=value>\
```

config:HostPifReadThresholdCritical=*<HostPifReadThresholdCritical=value>*\ config:HostPifReadThresholdHigh=*<HostPifReadThresholdHigh=value>*\ config:set_host_configuration=*<true | false>* ...\ ]

Use the **pool-send-wlb-configuration** command with the arguments *<ParticipatesInPowerManagement>* and *<set_host_configuration>* to configure Workload Balancing's Host Power Management feature.

```
xe pool-send-wlb-configuration \
config:<host_21_> \
ParticipatesInPowerManagement=<true> \
config:set_host_configuration=<true>
```

Appendix B. Workload Balancing Service Commands

Service Commands

Run the following service commands on the Workload Balancing appliance. To do so, you must log in to the Workload Balancing virtual appliance.

Logging in to the Workload Balancing Virtual Appliance

Before you can run any service commands or edit the wlb.conf file, you must log in to the Workload Balancing virtual appliance. To do so, you must enter a user name and password. Unless you created additional user accounts on the virtual appliance, log in using the root user account. You specified this account when you ran Workload Balancing Configuration wizard (before you connected your pool to Workload Balancing). You can, optionally, use the Console tab in XenCenter to log in to the appliance.

To log in to the Workload Balancing virtual appliance

1. At the *<name-of-your-WLB-VPX>* login prompt, enter the account user name. For example, where wlb-vpx-pos-pool is the name of your Workload Balancing appliance:

   ```
   wlb-vpx-pos-pool login: root
   ```
2. At the Password prompt, enter the password for the account:

   ```
   wlb-vpx-pos-pool login: root
   ```

   > **Note:**
   >
   > To log off the Workload Balancing virtual appliance, simply type **logout** at the command prompt.

service workloadbalancing restart

Run the **service workloadbalancing start** command from anywhere in the Workload Balancing appliance to stop and then restart the Workload Balancing Data Collection, Web Service, and Data Analysis services.

service workloadbalancing start

Run the **service workloadbalancing start** command from anywhere in the Workload Balancing appliance to start the Workload Balancing Data Collection, Web Service, and Data Analysis services.

service workloadbalancing stop

Run the **service workloadbalancing stop** command from anywhere in the Workload Balancing appliance to stop the Workload Balancing Data Collection, Web Service, and Data Analysis services.

service workloadbalancing status

Run the **service workloadbalancing status** command from anywhere in the Workload Balancing appliance to determine the status of the Workload Balancing server. After you execute this command, the status of the three Workload Balancing services (the Web Service, Data Collection Service, and Data Analysis Service) is displayed.

Modifying the Workload Balancing configuration options

Many Workload Balancing configurations, such as the database and web-service configuration options, are stored in the wlb.conf file, a configuration file on the Workload Balancing virtual appliance.

To make it easier to modify the most commonly used options, Citrix provides a command, **wlbconfig**. Running the **wlbconfig** command on the Workload Balancing virtual appliance lets you rename the Workload Balancing user account, change its password, or change the PostgreSQL password. After you execute this command, the Workload Balancing services are restarted.

To run the wlbconfig command

- Run the following from the command prompt:

  ```
  wlbconfig
  ```

  The screen displays a series of questions guiding you through changing your Workload Balancing user name and password and the PostgreSQL password. Follow the questions on the screen to change these items.

  Important:

  Double-check any values you enter in the wlb.conf file: Workload Balancing does not validate values in the wlb.conf file. Consequently, if the configuration parameters you specify are not within the required range, Workload Balancing does not generate an error log.

Editing the Workload Balancing configuration file

You can modify Workload Balancing configuration options by editing the wlb.conf file, which is stored in /opt/citrix/wlb directory on the Workload Balancing virtual appliance. In general, Citrix does not recommend changing the settings in this file without guidance from Citrix; however, there are three categories of settings you can change if desired:

- **Workload Balancing account name and password**. It is easier to modify these by running the **wlbconfig** command.
- **Database password**. This can be modified using the wlb.conf file. However, Citrix recommends modifying it through the **wlbconfig** command since this command modifies the wlb.conf file and automatically updates the password in the database. If you choose to modify the wlb.conf file instead, you must run a query to update the database with the new password.
- **Database grooming parameters**. You can modify database grooming parameters, such as the database grooming interval, using this file by following the instructions in the database-management section of the *Workload Balancing Administrator's Guide*. However, Citrix recommends using caution if you do so.

For all other settings in the wlb.conf file, Citrix currently recommends leaving them at their default, unless Citrix instructed you to modify them.

To edit the wlb.conf file

1. Run the following from the command prompt on the Workload Balancing virtual appliance (using VI as an example):

   ```
   vi /opt/citrix/wlb/wlb.conf
   ```

   The screen displays several different sections of configuration options.
2. Modify the configuration options, and exit the editor.

   You do not need to restart Workload Balancing services after editing the wlb.conf file. The changes go into effect immediately after exiting the editor.

   Important:

Double-check any values you enter in the wlb.conf file: Workload Balancing does not validate values in the wlb.conf file. Consequently, if the configuration parameters you specify are not within the required range, Workload Balancing does not generate an error log.

Increasing the Detail in the Workload Balancing Log

The Workload Balancing log provides a list of events on the Workload Balancing virtual appliance, including actions for the analysis engine, database, and audit log. This log file is found in this location: /var/log/wlb/LogFile.log.

You can, if desired, increase the level of detail the Workload Balancing log provides. To do so, modify the Trace flags section of the Workload Balancing configuration file (wlb.conf), which is found in the following location: /opt/citrix/wlb/wlb.conf. Enter a **1** to enable logging for a specific trace and a **2** to disable logging. For example, to enable logging for the Analysis Engine trace, enter:

```
<AnalEngTrace>1</AnalEngTrace>
```

You may want to increase logging detail before reporting an issue to Citrix Technical Support or when troubleshooting.

Logging Option	Trace Flag	Benefit or Purpose
Analysis Engine Trace	AnalEngTrace	Logs details of the analysis engine calculations. Shows details of the decisions the analysis engine is making and potentially gain insight into the reasons Workload Balancing is not making recommendations.
Database Trace	DatabaseTrace	Logs details about database reads/writes. However, leaving this trace on will increase the log file size quickly.
Data Collection Trace	DataCollectionTrace	Logs the actions of retrieving metrics. This lets you see the metrics Workload Balancing is retrieving and inserting into the Workload Balancing data store. However, leaving this trace on will increase the log file size quickly.
Data Compaction Trace	DataCompactionTrace	Logs details about how many milliseconds it took to compact the metric data.
Data Event Trace	DataEventTrace	This trace provides details about events Workload Balancing catches from XenServer.
Data Grooming Trace	DataGroomingTrace	This trace provides details about the database grooming.
Data Metrics Trace	DataMetricsTrace	Logs details about the parsing of metric data. Leaving this trace on will increase the log-file size quickly.
Queue Managemen Trace	QueueManagementTrace	Logs details about data collection queue management processing. (This is for internal use.)

Logging Option	Trace Flag	Benefit or Purpose
Data Save Trace	DataSaveTrace	Logs details about the pool being saved to the database.
Score Host Trace	ScoreHostTrace	Logs details about how Workload Balancing is arriving at a score for a host. This trace shows the detailed scores generated by Workload Balancing when it calculates the star ratings for selecting optimal servers for VM placement.
Audit Log Trace	AuditLogTrace	Shows the action of the audit log data being captured and written. (This is generally only for internal use and does not provide information that is captured in the audit log.) However, leaving this trace on will increase the log file size quickly.
Scheduled Task Trace	ScheduledTaskTrace	Logs details about scheduled tasks. For example, if your scheduled mode changes or report subscriptions are not working, you might want to enable this trace to investigate the cause.
Web Service Trace	WlbWebServiceTrace	Logs details about the communication with the web-service interface.