XenServer™

XenServer 8

Contents

About XenServer 8

March 28, 2024

XenServer is a virtualization platform that allows organizations to create and manage virtualized server infrastructures. It is designed to optimize the delivery of Windows and Linux virtual machines, providing a robust and scalable solution for data center virtualization.

XenServer offers features such as live migration, snapshot and cloning capabilities, and resource pooling, allowing for efficient management of virtualized workloads. It provides a secure and high-performance environment for running applications and services, making it a suitable choice for businesses looking to streamline their server infrastructure.

Cloud Software Group (CSG) emphasizes XenServer's integration with Citrix products, creating a comprehensive virtualization and application delivery solution. We aim to enhance the flexibility, agility, and cost-effectiveness of IT operations through centralized management and efficient resource utilization. Overall, XenServer is positioned as a reliable virtualization solution for businesses seeking a powerful and versatile platform for their server environments.

Get XenServer 8 here.

> **Note:**
>
> If you previously used XenServer 8 as a preview, apply the latest set of updates to move seamlessly to the production-supported version.
>
> If you are using a Citrix Virtual Apps and Desktops license with XenServer, you must change to a XenServer Premium Edition license. For more information, see https://xenserver.com/buy. Existing Citrix Virtual Apps and Desktops customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

Is XenServer 8 for me?

XenServer 8 is the latest release of XenServer. It succeeds Citrix Hypervisor 8.2 Cumulative Update 1 and includes many new features. For more information about these features, see What's New.

Why choose XenServer 8?

- You want to try the latest features from XenServer and can regularly update your hosts and pools.

  One notable change between Citrix Hypervisor 8.2 Cumulative Update 1 and XenServer 8 is our move to delivering features and fixes on an ongoing basis through our frequent updates mechanism. For more information, see Update your XenServer hosts.

- You want to use Windows 11 VMs in your environment.

  XenServer 8 delivers support for Windows 11 VMs and vTPMs. For more information, see Windows VMs.

- You are a Citrix Virtual Apps and Desktops user and want to run your workload on a hypervisor that contains many features optimized to your environment.

  For more information about these features, see Using XenServer with Citrix products.

  To learn which versions of Citrix Virtual Apps and Desktops (MCS) and Citrix Provisioning (PVS) are supported with XenServer 8, see Supported Hypervisors for Citrix Virtual Apps and Desktops (MCS) and Citrix Provisioning (PVS).

  You can also benefit from a limited-time promotion that offers free XenServer licenses for Citrix customers. Learn more.

In any of these cases, XenServer 8 is for you. Get it here.

XenCenter

XenServer 8 requires the latest version of XenCenter, which has a version number of the form "XenCenter YYYY.x.x". Previous versions of XenCenter, such as XenCenter 8.2.x, are not supported with XenServer 8.

- Download the latest XenCenter
- See the documentation

XenCenter YYYY.x.x is fully supported with XenServer 8. XenCenter YYYY.x.x is not yet supported for production use with Citrix Hypervisor 8.2 CU1.

Frequent updates across the XenServer 8 lifecycle

With XenServer 8, frequent updates are made available to you in XenCenter, allowing you to benefit from a more efficient release process that delivers new features and bug fixes at a faster cadence than was previously possible. This feature enables you to experience the frequent update model for administering updates to your XenServer pools and hosts.

During its lifecycle, XenServer 8 provides a stream of frequent and easy-to-apply updates, which enable you to consume new features and bug fixes at the earliest possible juncture. You must apply all available updates periodically. As a result, the behavior and feature set in XenServer 8 can change.

Get started

Steps to use XenServer 8:

1. Get XenServer 8 from the XenServer downloads page.
2. Install the latest version of XenCenter.
3. Install or upgrade to XenServer 8.
4. Apply updates by using XenCenter.

What's new

March 28, 2024

Our goal is to deliver new features and product updates to XenServer 8 customers as soon as they are ready. New releases provide more value, so there's no reason to delay updates. Through the XenServer 8 release stream, we deliver updates incrementally in waves to help ensure product quality and maximize availability.

> **Note:**
>
> If you previously used XenServer 8 as a preview, apply the latest set of updates to move seamlessly to the production-supported version.
>
> If you are using a Citrix Virtual Apps and Desktops license with XenServer, you must change to a XenServer Premium Edition license. For more information, see https://xenserver.com/buy. Existing Citrix Virtual Apps and Desktops customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

XenServer is back

We are once again releasing our product under the XenServer brand. For more information, see the XenServer website.

As a part of this change, some of the other names and terms used in our product and our documentation are changing:

Old term	New term	Notes
Citrix Hypervisor	XenServer	

Old term	New term	Notes
XenServer version format `major_version.minor_version`	XenServer version format `major_version`	The XenServer version format has changed to only show the major version. Where previous releases were numbered “XenServer 7.6”, “Citrix Hypervisor 8.2”, and so on, this release and any future releases show only the major version, for example, “XenServer 8”. XenServer 8 is based on the same platform as Citrix Hypervisor 8.2 CU1 and so shares the same major version. However, XenServer 8 is the newer version of the product and contains the latest features and fixes.
XenCenter x.x.x	XenCenter YYYY.x.x	The XenCenter version format has changed to be independent of the XenServer version. The new version format for XenCenter is `year.major_version.minor_version`.
Citrix VM Tools (previously XenServer PV Tools)	XenServer VM Tools	
Pool master	Pool coordinator	The main host in a pool is now referred to as the pool coordinator in the documentation and in XenCenter. The older term is still in use in some xe CLI commands and in the management API.

Old term	New term	Notes
Pool slave	Pool supporter	The subordinate hosts in a pool are now referred to as the pool supporters or supporting hosts in the documentation and in XenCenter. The older term is still in use in some xe CLI commands and in the management API.
Master password	Main password	
Express Edition	Trial Edition	

Frequent and easy-to-apply updates

In XenServer 8, the way we release updates to you has changed. Frequent updates are made available, enabling you to benefit from a more efficient release process that delivers new features and bug fixes at a faster cadence than was previously possible. Use XenCenter or the xe CLI to apply these updates to your XenServer hosts and pools at a time that is convenient to you. For more information, see Update your XenServer hosts.

1. We make frequent updates available for XenServer 8 in our secure CDN.
2. In XenCenter, see when updates are available for your pool.
3. Using XenCenter or the xe CLI, initiate the process of applying updates to your XenServer pool.

For more information, see Apply updates by using XenCenter or Apply updates by using the xe CLI.

For a list of the latest updates available for your Early Access or Normal pools, see the following pages:

- Early Access channel updates
- Normal channel updates

These pages do not list all changes in the Early Access and Normal channels, just a subset. For the full set of changes available, see the information in the XenCenter **Updates** view.

Windows 11 and vTPM support

Windows 11 is now supported on XenServer.

This feature also includes support for vTPMs. You can create and attach a vTPM to a Windows 10 or Windows 11 VM. For more information, see Windows VMs.

The vTPM provides a TPM 2.0 compliant API to applications in the VM. TPM 1.2 is not supported.

Licensing changes

The licensing behavior in XenServer 8 is different to that in earlier versions of Citrix Hypervisor and XenServer. We've changed the requirements for Citrix customers, added a new edition, and made some features available to everyone.

For more information, see Licensing.

Citrix Virtual Apps and Desktops and Citrix DaaS

In XenServer 8, you must have a Premium Edition license to run your workloads on a XenServer pool or host. This is a change from previous versions of Citrix Hypervisor or XenServer, which enabled you to use your Citrix Virtual Apps and Desktops or Citrix DaaS license to get a free entitlement to XenServer.

For more information about getting a XenServer license, see https://xenserver.com/buy.

If you already use XenServer or Citrix Hypervisor to host your Citrix Virtual Apps and Desktops workloads, you can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

To learn which versions of Citrix Virtual Apps and Desktops (MCS) and Citrix Provisioning (PVS) are supported with XenServer 8, see Supported Hypervisors for Citrix Virtual Apps and Desktops (MCS) and Citrix Provisioning (PVS).

Trial Edition

You can now try XenServer 8 for free with Trial Edition. The Trial Edition lets you try Premium Edition features, but in a restricted size pool of up to 3 hosts. For more information about the different editions of XenServer, see XenServer editions.

Feature changes

The following features, which in previous versions were restricted to Premium Edition customers, are now also available with Standard Edition:

- Automated updates to the Windows VM drivers

- Automated updates to the Management Agent
- Live patching
- XenServer Conversion Manager

Monitor host and dom0 resources with NRPE

> **Note:**
>
> The NRPE feature is available for XenServer Premium or Trial Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.

In XenServer 8, you can use any third-party monitoring tool that supports Nagios Remote Plugin Executor (NRPE) to monitor host and dom0 resources, such as Nagios Core. XenServer integrates NRPE into dom0, enabling you to capture various host and dom0 metrics. For more information, see Monitor host and dom0 resources with NRPE.

Monitor host and dom0 resources with SNMP

> **Note:**
>
> The SNMP feature is available for XenServer Premium or Trial Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.

You can now use SNMP and any NMS of your choosing to remotely monitor resources consumed by XenServer. With this feature, you can also configure traps to monitor your XenServer hosts, which are agent-initiated messages that alert the NMS that a specific event has occurred in XenServer. For more information, see Monitoring host and dom0 resources with SNMP.

Local XFS

You can now use local storage devices with 4 KB physical blocks without needing a logical block size of 512 bytes by using the new thin-provisioned local SR type: XFS. For more information, see Local XFS.

Changes to guest operating system support

For the full list of supported guest operating systems in XenServer 8, see Guest operating system support.

Added

XenServer 8 now supports the following new guests:

- Debian Bullseye 11 (64-bit)
- Ubuntu 22.04 (64-bit)
- Windows 11 (64-bit)

Removed

XenServer 8 no longer supports the following guests:

- Debian Jessie 8 (32-bit)
- Debian Jessie 8 (64-bit)
- Debian Stretch 9 (32-bit)
- Debian Stretch 9 (64-bit)
- Ubuntu 16.04 (32-bit)
- Ubuntu 16.04 (64-bit)
- CoreOS
- SUSE Linux Enterprise Desktop 12 SP3, 12 SP4 (64-bit)
- SUSE Linux Enterprise Desktop 15 SP3 (64-bit)
- SUSE Linux Enterprise Server 12 SP3 (64-bit)
- CentOS 8 (64-bit)
- Windows 10 (32-bit)

Deprecated

The following guests are deprecated in XenServer 8:

- Ubuntu 18.04 (64-bit)
- SUSE Linux Enterprise Desktop 12 SP4 (64-bit)

XenServer Conversion Manager 8.3.1

XenServer Conversion Manager 8.3.1 - the latest version of the XenServer Conversion Manager virtual appliance - allows you to convert VMs in parallel, enabling you to migrate your entire VMware environment to XenServer quickly and efficiently. You can convert up to 10 VMware ESXi/vCenter VMs at the same time.

For more information on how to use the XenServer Conversion Manager, see XenServer Conversion Manager.

Improvements to GFS2

Some restrictions on using GFS2 SRs with Citrix Machine Creation Services have been removed.

- You can now use MCS full clone VMs with GFS2 SRs.
- You can now use multiple GFS2 SRs in the same MCS catalog.

For more information about using GFS2 SRs, see Thin-provisioned shared GFS2 block storage.

Certificate verification

The certificate verification feature ensures that all TLS communication endpoints on the management network verify the certificates used to identity their peers before transmitting confidential data.

Certificate verification is enabled by default on fresh installations of XenServer 8 and later. If you upgrade from an earlier version of XenServer or Citrix Hypervisor, certificate verification is not enabled automatically and you must enable it. XenCenter prompts you to enable certificate verification the next time you connect to the upgraded pool.

For more information, see Certificate verification.

Restrict use of port 80

To improve security, XenServer 8 now allows you to close TCP port 80 on the management interface and exclusively use HTTPS over port 443 to communicate with XenServer. However, before closing port 80, check whether all your API clients (Citrix Virtual Apps and Desktops in particular) can use HTTPS over port 443.

By default, port 80 is still open. However, all internal connections for VM migration now use HTTPS over port 443 by default.

For more information about how to close port 80, see Restrict use of port 80.

Migration stream compression

The migration stream compression feature enables you to speed up the memory transfer on slow networks when live migrating a VM by compressing the data stream between the hosts. Enable it from XenCenter or the xe CLI. For more information, see Pool Properties - Advanced and Pool parameters.

Winbind replaces PBIS

Winbind has replaced PBIS for authenticating Active Directory (AD) users with the AD server and encrypting communications with the AD server. This replacement happens automatically when you upgrade. In the unlikely case that external authentication does not work after you upgrade to XenServer 8, leave the AD domain and rejoin it.

There are some minor differences in behavior as a result of this change:

- When using the command `xe pool-enable-external-auth` to join a domain, the parameter `config:disable_modules` is now ignored. This parameter is specific to PBIS.
- For the command `xe pool-enable-external-auth`, the parameter `config:ou` now supports either of the following formats when specifying a multiple layer OU: `config:ou=a/b/c` or `config:ou=c,ou=b,ou=a`.
- Winbind automatically updates the machine account password every 14 days or as specified by the configuration option `winbind_machine_pwd_timeout`.
- Winbind does not support the following scenarios:
  - Space at the beginning or end of a domain user or domain group name.
  - Domain user names that contain 64 characters or more.
  - Domain user names that include any of the special characters +<>"=/%@:,;'
  - Domain group names that include any of the special characters ,;'

For more information, see Winbind.

Integrated PVS-Accelerator

In previous releases of XenServer or Citrix Hypervisor, the PVS-Accelerator was provided as a supplemental pack. The PVS-Accelerator is now included in the base XenServer installation. The behavior of the PVS-Accelerator is otherwise unchanged and you must configure it before use.

For more information about the PVS-Accelerator, see PVS-Accelerator.

Network boot a VM over IPv6

You can now network boot a VM over an IPv6 network. This feature is only supported for UEFI VMs, not BIOS VMs.

Removed features

The following features are no longer supported in XenServer:

- Legacy partition layout
- Health Check
- Measured Boot Supplemental Pack
- Demo Linux virtual appliance

> **Note:**
>
> Logs for the Health Check service are retained by Windows for troubleshooting purposes. To remove these logs, delete them manually from `%SystemRoot%\System32\Winevt\Logs` on the Windows machine running XenCenter.

Changes to third-party components

PuTTY is no longer bundled with XenCenter. To launch an SSH console to a XenServer host by using XenCenter, you must install an external SSH console tool and ensure that XenCenter is configured to use it. For more information, see Configure XenCenter to use an external SSH console.

The following Broadcom binaries are no longer included in the XenServer installation:

- elxocmcore
- elxocmcorelibs
- hbaapiwrapper

To download these binaries from the Broadcom Emulex download page, complete the following steps:

1. Go to the **Management Software & Tools** section.
2. Download the **Emulex HBA Manager Core Application Kit (CLI) for Citrix XenServer**.

The following Marvell command-line binaries are no longer included in the XenServer installation:

- QConvergeConsole CLI for Citrix (QCC)
- QCS

To download QCC from the Marvell QLogic download page, complete the following steps:

1. Select the **Adapters** tab.
2. In the left panel, choose the type of adapter.
3. In the middle panel, choose the model of your adapter.
4. In the right panel, choose **Citrix Hypervisor**.
5. Click **Go**. You are redirected to a page with the available downloads.
6. Download the **QConvergeConsole CLI for Citrix (QCC)**.

To install this application, follow the instructions in the QConvergeConsole Command Line Utility User's Guide.

Compatibility notes

XenServer 8 is compatible with the following components:

- The latest version of the XenServer VM Tools for Windows
- The latest version of the XenServer VM Tools for Linux
- The latest version of the Workload Balancing virtual appliance
- The latest version of the XenServer Conversion Manager virtual appliance

These components are available on the XenServer Downloads page.

Early Access channel updates

May 7, 2024

The following features, preview features, improvements, and bug fixes are available in the Early Access update channel. Some of the latest listed entries might not be available in the Normal channel yet.

> **Note**
>
> This article doesn't list all changes in the Early Access channel, just a subset. For the full and up-to-date set of changes available, see the information in the XenCenter **Updates** view.

Apr 29, 2024

Checksum: aad9014d7a618f7cba0332b9bc4c95f9faee096c6495efb5d31d222a017e7c5b

These updates contain the following feature:

- These guest templates now support UEFI and Secure Boot:
  - Rocky Linux 8
  - Rocky Linux 9 (preview)
  - SUSE Linux Enterprise 15
  - Debian Bookworm 12 (preview)
  - Oracle Linux 8

These updates fix the following issues:

- Under some conditions DLM control daemon will fail to join a lockspace, preventing cluster operations from functioning.
- Some devices that are not intended for use for LVM volumes were not excluded from being scanned, which can cause operations to fail.

- Post-install warning on pvsproxy.

These updates include the following improvements:

- Add xsconsole "Configure Network Time"option for NTP control.

Apr 26, 2024

Checksum: c11e016b345e5119adb4f9edaffe3537cbf64ed252e3e7caebd30b380ec2a8b4

These updates fix the following issue:

- Enabling PVS-Accelerator in-memory cache mode fails with a script error.

Apr 11, 2024

Checksum: dc7acd63c2c5a920e77328070d7396dcb1df03ba670424fae3f4a29f47ff1a64

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/article/CTX633151.

Apr 02, 2024

Checksum: bbe72f61afa19b78c5f47fa59c2b3a527b018029bf34aa92ba3e9e4e35a14a46

These updates fix the following issues:

- Sometimes, when XAPI attempts to write logs, it is prevented from doing so for a limited amount of time. This behavior is caused by an issue in log rotation.
- If you create a new installation of XenServer on a host with a local XFS SR on an NVMe device, your local storage does not attach on boot. The action fails with the error: "Raised Server_error(SR_BACKEND_FAILURE, [FileNotFoundError; [Errno 2] No such file or directory: '/sys/block/nvme0n/queue/scheduler'])". After applying this fix, you can attach the local storage manually.

These updates include the following improvements:

- Improve the configuration for USB network cards.
- Improvements to plug/unplug behaviour on GFS2 and XFS SRs.
- Improvements to USB device handling.
- Addition of distributed tracing information to the storage manager.

Mar 25, 2024

Checksum: 574edaa4f5fe4960882e3a7ccd3de3084df6b89624dfa63fb7253e73120f1a41

These updates include the following improvements:

- Reduce QEMU's idle CPU usage.
- Update the Cisco enic driver to 4.5.0.7.
- Update the Cisco fnic driver to 2.0.0.90.

Mar 18, 2024

Checksum: e7dd9deb95cad01b70bac2d02aec7fa0d649c16b55cdd91d32affa6377631bae

> **Important:**
>
> Update to this level, or later, to be in a supported state.

These updates fix the following issue:

- After upgrading from Citrix Hypervisor 8.2 CU1, the background maintenance services for GFS2 SRs do not start correctly.

Mar 14, 2024

CHecksum: e8fe79a0d028b40a090a2d77d24d05caaa0fd5116080c7e3aaaed14458e60ede

These updates fix the following issues:

- If the pool coordinator is not running or the toolstack is being restarted, vTPM operations performed by the user or by Windows in the background might fail.
- Fix the names of some SR types in xsconsole.
- Upstream code changes that may reduce false-positive reports for CVE-2023-38545.

These updates include the following improvements:

- Upstream code changes that may reduce false-positive reports for CVE-2023-28486.

Mar 12, 2024

Checksum: b43aeaa6613a4b89d3d907cdb3704e0aca00c4d1122a01f1c4abac116602c2e4

These updates include the following changes to XenServer:

- If you use XenServer 8 preview with a Citrix Virtual Apps and Desktops license, this license is deprecated and is no longer supported with XenServer 8.

  To run a Citrix Virtual Apps and Desktops workload on a XenServer 8 pool, you must get a XenServer Premium Edition license for all hosts in the pool. For more information, visit the XenServer website.

  Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

These updates fix the following issue:

- XSA-452 CVE-2023-28746

  For more information, see the Security Bulletin: https://support.citrix.com/article/CTX616982.

These updates include the following improvements:

- Update the Intel IPU 2024.1 microcode release.

Mar 06, 2024

Checksum: 541fac9e361504d8c568fb6c5bbdef2442d5674ea5e17a2f1cf999c51e5b0608

> **Note:**
>
> Before applying these updates, update your XenCenter to version 2024.1.0 or later. The latest version of XenCenter is available at https://xenserver.com/downloads.

These updates contain the following feature:

- The "preview"label has been removed from the Windows 11 template. This guest operating system is ready to be fully supported when XenServer 8 moves from preview to fully supported in production.

These updates fix the following issues:

- When collecting host and guest performance statistics, the collected RRD files are often not up-to-date.
- In XenServer 8 clusters with GFS2 SRs, when collecting the XenServer databases, the cluster daemon database is not collected.
- When collecting a server status report, if you request a full bug-report archive and the file size limit of a database for xcp-rrdd-plugins is exceeded, the logfiles of xcp-rrdd-plugin are not collected.
- When collecting a server status report, if you use SSH login:

    - The interactive mode where you confirm individual files for collection does not work without specific user input.
    - When downloading uncompressed RRD data with a deprecated method, the VM RRDs are not collected.

- CVE-2023-45230 - Buffer overflow in the DHCPv6 client via a long Server ID option.
- CVE-2023-45231 - Out of Bounds read when handling a ND Redirect message with truncated options.
- CVE-2023-45232 - Infinite loop when parsing unknown options in the Destination Options header.
- CVE-2023-45233 - Infinite loop when parsing a PadN option in the Destination Options header.
- CVE-2023-45234 - Buffer overflow when processing DNS Servers option in a DHCPv6 Advertise message.
- CVE-2023-45235 - Buffer overflow when handling Server ID option from a DHCPv6 proxy Advertise message.
- An issue with the use of vTPM while the toolstack is being restarted.
- The status of NIC bonds is not reflected correctly.

These updates include the following improvements:

- Improvements to the way you apply software updates to your XenServer hosts and pools. For more information, see Apply updates.
- Update the multipath configuration used for PURE FlashArray SAN to match the vendors recommendations.
- Improve error messages when file SR types are read-only.
- Improvements to distributed tracing.

Feb 28, 2024

Checksum: 57b11c890cc12413b41310e4aef70b10589b50663ba1f5371e3a70c46bd7fa4d

These updates fix the following issues:

- XSA-451 CVE-2023-46841.
- A migration issue with VMs that previously saw CMP_LEGACY.

Feb 21, 2024

Checksum: 4ac84bc81dcd650b1525d7a1866363e6ff57988fd23b15f01f4db98a1da29984

These updates contain the following new feature:

- Monitor host and dom0 resources with SNMP. This feature can be used from the next version of XenCenter. For more information, see Monitoring host and dom0 resources with SNMP.

These updates fix the following issues:

- If you try to enable SR-IOV on an Intel E810 NIC in XenCenter, the VF assigned to a VM does not work after the VM boots.

These updates include the following improvements:

- Update Xen from 4.13 to 4.17.
- Various improvements to GFS2 datapath operations.

Feb 12, 2024

Checksum: 4ca53a5e440cd52b4b179b2dfb208e159cf16e918f53130b227eae3fb7b3d596

These updates include the following new feature:

- Add a new check (`check_multipath`) to the NRPE service to enable monitoring of the multipath status.

These updates fix the following issues:

- When there are separate independent Target Portal Groups within the IQN, XenServer cannot log into all iSCSI portals.
- When creating an SMB ISO SR share, it is no longer necessary to supply credentials when connecting to an SMB server that permits guest access.
- Part of the toolstack may unexpectedly stop running.
- You are unable to import a suspended Windows 11 VM that has been exported as an XVA with power state preservation.
- When a VM with vTPM is started or quickly migrated back and forth between pools, a race condition can occur.
- `pool-eject` operations run in parallel can cause TLS verification errors.
- Fixes for several low-probability issues with GFS2 SRs.

These updates include the following improvements:

- Update the Qlogic fastlinq driver to 8.74.0.2.
- Improvements to distributed tracing.

Jan 29, 2024

Checksum: bef04584f387457b7665a7f56b3f01be4c03edf67c383875ecbc8ebe92b73786

These updates include the following improvements:

- Support for UEFI boot and Secure Boot for Linux guest operating systems. For more information, see Guest UEFI boot and Secure Boot.
- Provide more detailed information about a XenServer host's SHA256 and SHA1 TLS certificates in the host console view in XenCenter.

Jan 23, 2024

Checksum: a74ad78f7b0537f82cb5069c6b781696829f4053b7a06020ab1436332f684ff1

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/arti

Jan 15, 2024

Checksum: 1633ab16bf0a6b458c4f863b07328e07ee85371fed34145050552c02e46657da

These updates include the following improvements:

- Update XenServer VM Tools for Linux to version 8.4.0-1, available to download from the XenServer downloads page. From this version, you can use the `install.sh` script to uninstall XenServer VM Tools for Linux. For more information, see Uninstall XenServer VM Tools for Linux.
- Ensure that the XenServer welcome message is always displayed when connecting to a XenServer host console and that the message is correctly line wrapped.

Jan 4, 2024

Checksum: 6240ec0cbc29f6f5204f7f6af7bbce28e96e9dfb9e001a4f0cfa4609214f9d8e

These updates fix the following issues:

- Sometimes, after migrating a VM from one pool to another, the VM's alerts are not successfully copied to the destination pool or removed from the source pool.
- Sometimes the pool database is not restored from the redo-log (part of the high availability feature).

These updates include the following improvements:

- Update the Microsemi smartpqi driver to 2.1.26_030.
- Update the AMD microcode to the 2023-12-05 drop.
- Improvements to distributed tracing.

Dec 11, 2023

Checksum: 95f70cca5fd3b79081c30837352ed941a7d497899062d892fbc2aa51daacfd78

These updates fix the following issues:

- When using multipathing with Dell EqualLogic PS Series Firmware v7.x, you might see iSCSI protocol errors.

Nov 27, 2023

Checksum: 1633ab16bf0a6b458c4f863b07328e07ee85371fed34145050552c02e46657da

These updates fix the following issues:

- Sometimes the redo-log (part of the high availability feature) does not replay all database writes.
- When running the `vdi-copy` command to copy a VDI to an SR, XenServer fails to correctly report the progress of the operation.
- In XenCenter, customizing a template and then exporting and reimporting it can result in the template failing to import.

These updates include the following improvements:

- Add driver for Dell PERC12 (driver version: mpi3mr 8.1.4.0.0).
- General API improvements.
- The "preview"label has been removed from vTPM support. This means that vTPM is ready to be fully supported when XenServer 8 moves from preview to fully supported in production.

Nov 15, 2023

Checksum: 4abd37e1e88675bf3793b4704786a97ba43e2a455989546de07f934f20e28135

These updates include the following improvement:

- The "preview"label has been removed from the Ubuntu 22.04 template. This guest operating system is ready to be fully supported when XenServer 8 moves from preview to fully supported in production.

Nov 14, 2023

Checksum: 052f8594042ceeb93bbe7d4dd75fc7fcd8af7092c3cd08b2522b6bd2a6694c17

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/article/CTX583037/.

Nov 6, 2023

Checksum: 210d00c5764f4d8ee0337ef827863eb8cde43fe089d888dd3faa2ee5d05a24ad

These updates include the following improvements:

- General SDK improvements and improved API login times.
- Update Open vSwitch to v2.17.7.

Nov 3, 2023

Checksum: 4d516d4b6c72eb3be4f60e78371db5fde8b6292395ddaf411faf52507f22f516

These updates include general fixes and improvements.

Oct 25, 2023

Checksum: c6b1397dc454e7236634e07f7872ec95e3f6938bcb5dc62316dd112899926ec7

These updates fix the following issues:

- Deploying Windows using PXE boot and Configuration Manager can cause Windows to hang.
- If time synchronization is disabled, Windows VMs do not return the correct time.

These updates include the following improvement:

- UEFI boot mode Windows VMs now show the Windows logo during boot instead of the Tianocore logo.

Oct 18, 2023

Checksum: 85e8edbaf7469a29cc56fe6880e93c2618fb90002d2d4cdf0b372719413c4525

These updates include general fixes and improvements.

Oct 11, 2023

Checksum: 7623587c09c7198237bc9dd673fba33307fbaee18e0baec50c1e2ddbee165b91

Support for the following guest operating systems:

- Debian Bookworm 12
- Rocky Linux 9
- CentOS Stream 9

> **Note:**
>
> Customers who wish to use these guest operating systems must also install XenServer VM Tools for Linux v8.3.1-1 or later, available to download from the XenServer product downloads page.

These updates include the following improvement:

- Update the Mellanox mlnx_en driver to 5.9-0.5.5.0.

Oct 10, 2023

Checksum: 0ab01aa0c5623b52219b279b48f6b13f8db4a0a699a25b61a7aa35b39243ca89

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/article/CTX575089/.

Oct 2, 2023

These updates include the following improvement:

- Support for the Red Hat Enterprise Linux 9 operating system. Refer to the Red Hat Enterprise Linux 9 release notes for further technical information.

> **Note:**
>
> Customers who wish to use this guest OS must also install Citrix VM Tools for Linux v8.3.1-1 or later, available to download from the XenServer product downloads page.

Sep 18, 2023

These updates include general fixes and improvements.

Sep 11, 2023

These updates include the following improvements:

- Enable interrupt balancing for Fibre Channel (FC) PCI devices. This improves performance on fast FC HBA SRs, especially if multipathing is used.
- Fix for AMD errata #1474. Disable C6 after 1000 days of uptime on AMD Zen2 systems to avoid a crash at ~1044 days.

Aug 31, 2023

These updates fix the following issues:

- Performance metrics are not available for GFS2 SRs and disks on these SRs.
- You cannot snapshot or checkpoint a suspended VM if that VM has a vTPM attached.
- If a XenServer host crashes, or shuts down abruptly, starting more Windows 11 VMs or migrating more Windows 11 VMs onto that host eventually fails.
- If, when iSCSI SRs are attached, not all the possible paths are available (for example, an offline controller remote port is down), the SR does not have additional iSCSI sessions added when those remote ports are accessible again.

These updates include the following improvements:

- Update the Intel ice driver to v1.11.17.1.
- Performance improvements for GFS2.

Normal channel updates

May 7, 2024

Updates progress from the Early Access update channel to Normal on a regular cadence. The following features, preview features, improvements, and bug fixes are available in the Normal update channel.

> **Note:**
>
> This article doesn't list all changes in the Normal channel, just a subset. For the full and up-to-date set of changes available, see the information in the XenCenter **Updates** view.

May 07, 2024

Checksum: aad9014d7a618f7cba0332b9bc4c95f9faee096c6495efb5d31d222a017e7c5b

These updates contain the following feature:

- These guest templates now support UEFI and Secure Boot:
  - Rocky Linux 8
  - Rocky Linux 9 (preview)
  - SUSE Linux Enterprise 15
  - Debian Bookworm 12 (preview)

    - Oracle Linux 8

These updates fix the following issues:

- Under some conditions DLM control daemon will fail to join a lockspace, preventing cluster operations from functioning.
- Some devices that are not intended for use for LVM volumes were not excluded from being scanned, which can cause operations to fail.
- Post-install warning on pvsproxy.

These updates include the following improvements:

- Add xsconsole "Configure Network Time"option for NTP control.

Apr 26, 2024

Checksum: c11e016b345e5119adb4f9edaffe3537cbf64ed252e3e7caebd30b380ec2a8b4

These updates fix the following issue:

- Enabling PVS-Accelerator in-memory cache mode fails with a script error.

Apr 16, 2024

Checksum: dc7acd63c2c5a920e77328070d7396dcb1df03ba670424fae3f4a29f47ff1a64

These updates fix the following issues:

- Sometimes, when XAPI attempts to write logs, it is prevented from doing so for a limited amount of time. This behavior is caused by an issue in log rotation.
- If you create a new installation of XenServer on a host with a local XFS SR on an NVMe device, your local storage does not attach on boot. The action fails with the error: "Raised Server_error(SR_BACKEND_FAILURE, [FileNotFoundError; [Errno 2] No such file or directory: '/sys/block/nvme0n/queue/scheduler'])". After applying this fix, you can attach the local storage manually.

These updates include the following improvements:

- Improve the configuration for USB network cards.
- Improvements to plug/unplug behaviour on GFS2 and XFS SRs.
- Improvements to USB device handling.
- Addition of distributed tracing information to the storage manager.

Apr 11, 2024

Checksum: 617080b043b1c3766f9ffb5183616cbf7117d78c2631317fa614ec4699213e5e

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/article/CTX633151.

Apr 02, 2024

Checksum: 574edaa4f5fe4960882e3a7ccd3de3084df6b89624dfa63fb7253e73120f1a41

These updates include the following improvements:

- Reduce QEMU's idle CPU usage.
- Update the Cisco enic driver to 4.5.0.7.
- Update the Cisco fnic driver to 2.0.0.90.

Mar 19, 2024

Checksum: e7dd9deb95cad01b70bac2d02aec7fa0d649c16b55cdd91d32affa6377631bae

These updates fix the following issues:

- Fix the names of some SR types in xsconsole.
- Upstream code changes that may reduce false-positive reports for CVE-2023-38545.

These updates include the following improvements:

- Upstream code changes that may reduce false-positive reports for CVE-2023-28486.

Mar 18, 2024

Checksum: 49b11e4371d94e6edbc3f39d1a9944efe082c870587642903de0b734caf3896b

> **Important:**
>
> Update to this level, or later, to be in a supported state.

These updates fix the following issues:

- If the pool coordinator is not running or the toolstack is being restarted, vTPM operations performed by the user or by Windows in the background might fail.
- After upgrading from Citrix Hypervisor 8.2 CU1, the background maintenance services for GFS2 SRs do not start correctly.

Mar 12, 2024

Checksum: b43aeaa6613a4b89d3d907cdb3704e0aca00c4d1122a01f1c4abac116602c2e4

These updates include the following changes to XenServer:

- If you use XenServer 8 preview with a Citrix Virtual Apps and Desktops license, this license is deprecated and is no longer supported with XenServer 8.

  To run a Citrix Virtual Apps and Desktops workload on a XenServer 8 pool, you must get a XenServer Premium Edition license for all hosts in the pool. For more information, visit the XenServer website.

  Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

These updates fix the following issue:

- XSA-452 CVE-2023-28746

  For more information, see the Security Bulletin: https://support.citrix.com/article/CTX616982.

These updates include the following improvements:

- Update the Intel IPU 2024.1 microcode release.

Mar 07, 2024

Checksum: 541fac9e361504d8c568fb6c5bbdef2442d5674ea5e17a2f1cf999c51e5b0608

> **Note:**
>
> Before applying these updates, update your XenCenter to version 2024.1.0 or later. The latest version of XenCenter is available at https://xenserver.com/downloads.

These updates contain the following feature:

- The "preview"label has been removed from the Windows 11 template. This guest operating system is ready to be fully supported when XenServer 8 moves from preview to fully supported in production.

These updates fix the following issues:

- When collecting host and guest performance statistics, the collected RRD files are often not up-to-date.
- In XenServer 8 clusters with GFS2 SRs, when collecting the XenServer databases, the cluster daemon database is not collected.

- When collecting a server status report, if you request a full bug-report archive and the file size limit of a database for xcp-rrdd-plugins is exceeded, the logfiles of xcp-rrdd-plugin are not collected.
- When collecting a server status report, if you use SSH login:
  - The interactive mode where you confirm individual files for collection does not work without specific user input.
  - When downloading uncompressed RRD data with a deprecated method, the VM RRDs are not collected.
- CVE-2023-45230 - Buffer overflow in the DHCPv6 client via a long Server ID option.
- CVE-2023-45231 - Out of Bounds read when handling a ND Redirect message with truncated options.
- CVE-2023-45232 - Infinite loop when parsing unknown options in the Destination Options header.
- CVE-2023-45233 - Infinite loop when parsing a PadN option in the Destination Options header.
- CVE-2023-45234 - Buffer overflow when processing DNS Servers option in a DHCPv6 Advertise message.
- CVE-2023-45235 - Buffer overflow when handling Server ID option from a DHCPv6 proxy Advertise message.
- An issue with the use of vTPM while the toolstack is being restarted.
- The status of NIC bonds is not reflected correctly.

These updates include the following improvements:

- Improvements to the way you apply software updates to your XenServer hosts and pools. For more information, see Apply updates.
- Update the multipath configuration used for PURE FlashArray SAN to match the vendors recommendations.
- Improve error messages when file SR types are read-only.
- Improvements to distributed tracing.

Feb 28, 2024

Checksum: 57b11c890cc12413b41310e4aef70b10589b50663ba1f5371e3a70c46bd7fa4d

These updates contain the following new feature:

- Monitor host and dom0 resources with SNMP. This feature can be used from the next version of XenCenter.

These updates fix the following issues:

- If you try to enable SR-IOV on an Intel E810 NIC in XenCenter, the VF assigned to a VM does not work after the VM boots.
- XSA-451 CVE-2023-46841.
- A migration issue with VMs that previously saw CMP_LEGACY.

These updates include the following improvements:

- Update Xen from 4.13 to 4.17.
- Various improvements to GFS2 datapath operations.

Feb 15, 2024

Checksum: 4ca53a5e440cd52b4b179b2dfb208e159cf16e918f53130b227eae3fb7b3d596

These updates include the following new feature:

- Add a new check (`check_multipath`) to the NRPE service to enable monitoring of the multipath status.

These updates fix the following issues:

- When there are separate independent Target Portal Groups within the IQN, XenServer cannot log into all iSCSI portals.
- When creating an SMB ISO SR share, it is no longer necessary to supply credentials when connecting to an SMB server that permits guest access.
- Part of the toolstack may unexpectedly stop running.
- You are unable to import a suspended Windows 11 VM that has been exported as an XVA with power state preservation.
- When a VM with vTPM is started or quickly migrated back and forth between pools, a race condition can occur.
- `pool-eject` operations run in parallel can cause TLS verification errors.
- Fixes for several low-probability issues with GFS2 SRs.

These updates include the following improvements:

- Update the Qlogic fastlinq driver to 8.74.0.2.
- Improvements to distributed tracing.

Feb 01, 2024

Checksum: bef04584f387457b7665a7f56b3f01be4c03edf67c383875ecbc8ebe92b73786

These updates include the following improvements:

- Support for UEFI boot and Secure Boot for Linux guest operating systems. For more information, see Guest UEFI boot and Secure Boot.
- Provide more detailed information about a XenServer host's SHA256 and SHA1 TLS certificates in the host console view in XenCenter.

Jan 23, 2024

Checksum: a74ad78f7b0537f82cb5069c6b781696829f4053b7a06020ab1436332f684ff1

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/arti

Jan 18, 2024

Checksum: 1633ab16bf0a6b458c4f863b07328e07ee85371fed34145050552c02e46657da

These updates include the following improvements:

- Update XenServer VM Tools for Linux to version 8.4.0-1, available to download from the XenServer downloads page. From this version, you can use the `install.sh` script to uninstall XenServer VM Tools for Linux. For more information, see Uninstall XenServer VM Tools for Linux.
- Ensure that the XenServer welcome message is always displayed when connecting to a XenServer host console and that the message is correctly line wrapped.

Jan 8, 2024

Checksum: 6240ec0cbc29f6f5204f7f6af7bbce28e96e9dfb9e001a4f0cfa4609214f9d8e

These updates fix the following issues:

- Sometimes, after migrating a VM from one pool to another, the VM's alerts are not successfully copied to the destination pool or removed from the source pool.
- Sometimes the pool database is not restored from the redo-log (part of the high availability feature).

These updates include the following improvements:

- Update the Microsemi smartpqi driver to 2.1.26_030.
- Update the AMD microcode to the 2023-12-05 drop.
- Improvements to distributed tracing.

Dec 14, 2023

Checksum: 95f70cca5fd3b79081c30837352ed941a7d497899062d892fbc2aa51daacfd78

These updates fix the following issues:

- When using multipathing with Dell EqualLogic PS Series Firmware v7.x, you might see iSCSI protocol errors.

Nov 30, 2023

Checksum: 1633ab16bf0a6b458c4f863b07328e07ee85371fed34145050552c02e46657da

These updates fix the following issues:

- Sometimes the redo-log (part of the high availability feature) does not replay all database writes.
- When running the `vdi-copy` command to copy a VDI to an SR, XenServer fails to correctly report the progress of the operation.
- In XenCenter, customizing a template and then exporting and reimporting it can result in the template failing to import.

These updates include the following improvements:

- Add driver for Dell PERC12 (driver version: mpi3mr 8.1.4.0.0).
- General API improvements.
- The “preview”label has been removed from vTPM support. This means that vTPM is ready to be fully supported when XenServer 8 moves from preview to fully supported in production.

Nov 22, 2023

Checksum: 4abd37e1e88675bf3793b4704786a97ba43e2a455989546de07f934f20e28135

These updates include the following improvement:

- The “preview”label has been removed from the Ubuntu 22.04 template. This guest operating system is ready to be fully supported when XenServer 8 moves from preview to fully supported in production.

Nov 15, 2023

Checksum: 052f8594042ceeb93bbe7d4dd75fc7fcd8af7092c3cd08b2522b6bd2a6694c17

These updates include the following improvements:

- General SDK improvements and improved API login times.
- Update Open vSwitch to v2.17.7.

Nov 14, 2023

Checksum: 82dc14f34880f471e23467c1cc296bd7b9ca6a670ac71950e6db88019bb47a13

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/article/CTX583037/.

Nov 6, 2023

Checksum: 4d516d4b6c72eb3be4f60e78371db5fde8b6292395ddaf411faf52507f22f516

These updates fix the following issues:

- Deploying Windows using PXE boot and Configuration Manager can cause Windows to hang.
- If time synchronization is disabled, Windows VMs do not return the correct time.

These updates include the following improvement:

- UEFI boot mode Windows VMs now show the Windows logo during boot instead of the Tianocore logo.

Oct 23, 2023

Checksum: 85e8edbaf7469a29cc56fe6880e93c2618fb90002d2d4cdf0b372719413c4525

These updates include general fixes and improvements.

Oct 16, 2023

Checkcum: 7623587c09c7198237bc9dd673fba33307fbaee18e0baec50c1e2ddbee165b91

These updates include support for the following guest operating systems:

- Debian Bookworm 12
- Rocky Linux 9
- CentOS Stream 9

> **Note:**

> Customers who wish to use these guest operating systems must also install XenServer VM Tools for Linux v8.3.1-1 or later, available to download from the XenServer product downloads page.

These updates include the following improvement:

- Update the Mellanox mlnx_en driver to 5.9-0.5.5.0.

Oct 10, 2023

Checksum: 0ab01aa0c5623b52219b279b48f6b13f8db4a0a699a25b61a7aa35b39243ca89

These updates include security fixes. For more information, see the security bulletin https://support.citrix.com/article/CTX575089/.

Oct 5, 2023

Checksum: 7ef0331921b2d46b882efa76ea9d6a4c287868dd9a3b0eae4a6d15ba2cf29fbe

These updates include support for the Red Hat Enterprise Linux 9 operating system. Refer to the Red Hat Enterprise Linux 9 release notes for further technical information.

> **Note:**
>
> Customers who wish to use this guest OS must also install Citrix VM Tools for Linux v8.3.1-1 or later, available to download from the XenServer product downloads page.

Sep 21, 2023

These updates include general fixes and improvements.

Sep 14, 2023

These updates include the following improvements:

- Enable interrupt balancing for Fibre Channel (FC) PCI devices. This improves performance on fast FC HBA SRs, especially if multipathing is used.
- Fix for AMD errata #1474. Disable C6 after 1000 days of uptime on AMD Zen2 systems to avoid a crash at ~1044 days.

Sep 4, 2023

These updates fix the following issues:

- Performance metrics are not available for GFS2 SRs and disks on these SRs.
- You cannot snapshot or checkpoint a suspended VM if that VM has a vTPM attached.
- If a XenServer host crashes, or shuts down abruptly, starting more Windows 11 VMs or migrating more Windows 11 VMs onto that host eventually fails.
- If, when iSCSI SRs are attached, not all the possible paths are available (for example, an offline controller remote port is down), the SR does not have additional iSCSI sessions added when those remote ports are accessible again.

These updates include the following improvements:

- Update the Intel ice driver to v1.11.17.1.
- Performance improvements for GFS2.

Fixed issues

March 27, 2024

The following issues have been fixed in XenServer 8. In addition to these fixed issues, additional fixes are released on the Normal and Early Access update channels.

General

- Rebooting the VM does not have the same effect as powering off and then starting the VM. (CA-188042)
- If an Active Directory user inherits the pool admin role from an AD group that has spaces in its name, the user cannot log in to XenServer 8 through SSH. (CA-363207)
- In clustered pools, a network outage might cause the following issues: inability to reconnect to the GFS2 storage after a host reboot, inability to add or remove hosts in a pool, difficulties in managing the pool. (XSI-1386)

Graphics

- On hardware with NVIDIA A16/A2 graphics cards, VMs with vGPUs can sometimes fail to migrate with the internal error "Gpumon_interface.Gpumon_error([S(Internal_error);S((Failure "No vGPU available"))])". (CA-374118)

Guests

Windows guests

- When UEFI-boot Windows VMs start, they show a TianoCore logo. (CP-30146)
- On a Windows VM, sometimes the IP address of an SR-IOV VIF is not visible in XenCenter. (CA-340227)
- On a Windows VM with more than 8 vCPUs, Receive Side Scaling might not work because the xenvif driver fails to set up the indirection table. (CA-355277)
- If a XenServer host crashes, or shuts down abruptly, starting more Windows 11 VMs or migrating more Windows 11 VMs onto that host eventually fails. (CA-375992)

Linux guests

- The XenServer VM Tools for Linux can provide an incorrect value for the free memory of the VM that is higher than the correct value. (CA-352996)

Storage

- When attaching an iSCSI LVM SR with multi-target and wildcard targetIQNs to a host, the attach operation can fail if not all targets respond. (CA-375968)
- If a GFS2 SR has less than 500 MB of space on it, when you attempt to delete disks stored on this SR, the operation can fail. (CA-379589)
- When attempting to repair a connection to a read-only NFS v3 SR, the operation can fail on the first attempt with the error "SM has thrown a generic python exception". To work around this issue, attempt the repair operation again. This issue is caused by a write operation in the initial repair attempt. (XSI-1374)

Updates

- While the install update is in progress on a pool member, you might see the error "The operation could not be performed because getting updates is in progress.". To resolve this error, you can retry the operation. (CA-381215)

Workload Balancing

- For a Workload Balancing virtual appliance version 8.2.2 and later that doesn't use LVM, you cannot extend the available disk space. (CA-358817)

- In XenCenter, the date range showed on the Workload Balancing Pool Audit Report is incorrect. (CA-357115)
- During the Workload Balancing maintenance window, Workload Balancing is unable to provide placement recommendations. When this situation occurs, you see the error: "4010 Pool discovery has not been completed. Using original algorithm."The Workload Balancing maintenance window is less than 20 minutes long and by default is scheduled at midnight. (CA-359926)

XenCenter

For information about known and fixed issues in XenCenter, see XenCenter What's New.

Known issues

April 29, 2024

This article contains advisories and minor issues in the XenServer 8 release and any workarounds that you can apply.

General

- The backup and restore capability in xsconsole is temporarily not available. We are actively investigating a replacement solution. (CP-48776)
- When attempting to use the serial console to connect to a XenServer host, the serial console might refuse to accept keyboard input. If you wait until after the console refreshes twice, the console then accepts keyboard input. (CA-311613)
- When read caching is enabled, it is slower to read from the parent snapshot than from the leaf. (CP-32853)
- When attempting to log in to the dom0 console with an incorrect password, you receive the following error message: `When trying to update a password, this return status indicates that the value provided as the current password is not correct.` This error message is expected even though it relates to a password change, not a login. Try to log in with the correct password. (CA-356441)
- If a XenServer host is powered off unexpectedly and restarted, when it attempts to recover VMs that had an attached vTPM, the vTPM can sometimes be missing from the VM. (CA-379928)

Graphics

- When NVIDIA T4 added in pass-through mode to a VM on some specific server hardware, that VM might not power on. (CA-360450)

Guests

- If you attempt to live migrate a VM with dynamic memory control enabled to a target host where resources such as memory are very constrained, the migration can sometimes fail. (CA-380607)

Windows guests

- For domain-joined Windows 10 VMs (1903 and later) with FireEye Agent installed, repeated successful RDP connections can cause the VM to freeze with 100% CPU usage in `ntoskrnl.exe`. Perform a hard reboot on the VM to recover from this state. (CA-323760)
- When you create a UEFI VM, the Windows installation requires a key press to start. If you do not press a key during the required period, the VM console switches to the UEFI shell.

  To work around this issue, you can restart the installation process in one of the following ways:

  - In the UEFI console, type the following commands.

    ```
    EFI:
    EFI\BOOT\BOOTX64
    ```

  - Reboot the VM

  When the installation process restarts, watch the VM console for the installation prompt. When the prompt appears, press any key. (CA-333694)
- When attempting to update a Windows 10 VM from 1909 to 20H2 or later, the update might fail with a blue screen showing the error: INACCESSIBLE BOOT DEVICE. (XSI-1075)

  To make it less likely that this failure occurs, you can take the following steps before attempting to update:

  1. Update the XenServer VM Tools for Windows on your VM to the latest version.
  2. Snapshot the VM.
  3. In the VM registry, delete the following values from the HKEY_LOCAL_MACHINE\SYSTEM\CurrentContro key: ActiveDeviceID, ActiveInstanceID, and ActiveLocationInformation
- When creating a Windows VM from a template that is set to not automatically update its drivers, the created VM is incorrectly set to update its drivers. To work around this issue, run

the following command: `xe pool-param-set policy-no-vendor-device=true uuid=<pool-uuid>`. This command ensures that future VMs created from the template are correctly set to not automatically update drivers. VMs that were previously generated from the template are not changed. (CA-371529)

- vTPM operations performed by the user or by Windows in the background might fail in the following situations:

  - If the toolstack or the XenServer host crash before the operation is synchronized to the disk. Errors when writing to disk are ignored.

  In the case of this type of failure the vTPM returns an error to the operating system. Windows logs these errors to the System Event log.

- If you have Bitlocker enabled on a Windows VM and attempt to suspend or resume that VM, the VM can sometimes crash. (CA-368791)

  In addition, Bitlocker is not supported with VMs that have an attached vTPM.

Linux guests

- You cannot use the Dynamic Memory Control (DMC) feature on Red Hat Enterprise Linux 8, Red Hat Enterprise Linux 9, Rocky Linux 8, Rocky Linux 9, or CentOS Stream 9 VMs as these operating systems do not support memory ballooning with the Xen hypervisor. (CA-378797)

- On some Linux VMs, especially busy systems with outstanding disk I/O, attempts to suspend or live migrate the VM might fail. To work around this issue, try increasing the value of `/sys/power/pm_freeze_timeout`, for example, to 300000. If this work around is not successful, you can upgrade the Linux kernel of the VM to the latest version. (CP-41455)

- If you install Debian 10 (Buster) by using PXE network boot, do not add `console=tty0` to the boot parameters. This parameter can cause issues with the installation process. Use only `console=hvc0` in the boot parameters. (CA-329015)

- Due to a known issue with some SUSE Linux operating systems, if you attempt to trigger a crash dump on a SUSE Linux VM with 32 or more vCPUs, the operation fails and the VM does not restart automatically. This issue affects the following operating systems: SUSE Linux Enterprise Server 15 SP1, 15 SP2, 15 SP3, 15 SP4. (CA-375759)

- If you create a VM with RHEL 8.7 or earlier and only 1 vCPU, the VM times out while booting. To work around this issue, use RHEL 8.8 or later or change the number of vCPUs to be 2 or more. (CA-376921)

Installation

- When upgrading to or installing XenServer 8 from an ISO located on an IIS server, the install or upgrade can fail and leave your hosts unable to restart. The remote console shows the GRUB error: "File '/boot/grub/i3860pc/normal.mod'not found. Entering rescue mode". This issue is caused by the IIS configuration causing package files to be missing. To work around this issue, ensure that double escaping is allowed on IIS before extracting the installation ISO on it. (XSI-1063)
- Use the latest XenCenter to upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8. Using an older version of XenCenter can result in a loss of connectivity.

  Download the latest XenCenter from the XenServer product downloads page.

Internationalization

- Non-ASCII characters, for example, characters with accents, cannot be used in the host console. (CA-40845)
- In a Windows VM with XenServer VM Tools for Windows installed, copy and paste of double-byte characters can fail if using the default desktop console in XenCenter. The pasted characters appear as question marks (?).

  To work around this issue, you can use the remote desktop console instead. (CA-281807)

Storage

- If you create a new installation of XenServer on a host with a local XFS SR on an NVMe device, your local storage does not attach on boot. The action fails with the error: "Raised Server_error(SR_BACKEND_FAILURE, [FileNotFoundError; [Errno 2] No such file or directory: '/sys/block/nvme0n/queue/scheduler'])".

  After applying the latest updates, you can attach the local storage manually.
- If you use GFS2 SRs and have two hosts in your clustered pool, your cluster can lose quorum and fence during an upgrade. To avoid this situation, either add a host to or remove a host from your cluster. Ensure that you have either one or three hosts in your pool during the upgrade process. (CA-313222)
- If you are using a GFS2 SR and your cluster network is on a non-management VLAN, you cannot add or remove hosts to your clustered pool. (XSI-1604)
- After removing an HBA LUN from a SAN, you might see log messages and I/O failures when querying Logical Volume information. To work around this issue, reboot the XenServer host. (XSI-984)

- You cannot set or change the name of the tmpfs SR used by the PVS-Accelerator. When the `type` is `tmpfs`, the command `xe sr-create` disregards the value set for `name-label` and instead uses a fixed value. If you attempt to run the command `xe sr-param-set` to change the name of the tmpfs SR, you receive the error SCRIPT_MISSING.
- You cannot run more than 200 PVS-Accelerator-enabled VMs on a XenServer host. (CP-39386)

Third-party

- A limitation in recent SSH clients means that SSH does not work for usernames that contain any of the following characters: `{ } []|&`. Ensure that your usernames and Active Directory server names do not contain any of these characters.

XenCenter

For information about known and fixed issues in XenCenter, see XenCenter What's New.

Deprecation

April 25, 2024

The announcements in this article give you advanced notice of platforms, products, and features that are being phased out so that you can make timely business decisions. We monitor customer use and feedback to determine when they are withdrawn. Announcements can change in subsequent releases and might not include every deprecated feature or functionality.

For details about product lifecycle support, see the Product Lifecycle Support Policy article.

Deprecations and removals

The following table shows the platforms, products, and features that are deprecated or removed.

Deprecated items are not removed immediately. We continue to support them in the current release, but they will be removed in a future release.

Removed items are either removed, or are no longer supported, in XenServer.

Dates in **bold** face indicate changes at this release.

Item	Deprecation announced in	Removed in	Alternative
XenServer entitlement for Citrix Virtual Apps and Desktops customers	**XenServer 8**	**XenServer 8**	Get a Premium Edition license from https://xenserver.com/buy. Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.
BIOS boot mode for XenServer hosts	**XenServer 8**		Install your hosts with UEFI boot mode instead.
XenCenter connections to XenServer hosts that are version 7.x and earlier.	**XenCenter 2023.3.1**	**XenCenter 2023.3.1**	Upgrade your out-of-support XenServer hosts.
Demo Linux Virtual Appliance	**XenServer 8**	**XenServer 8**	
Support for 32-bit Windows 10	**XenServer 8**	**XenServer 8**	
Support for 32-bit Debian Bullseye 11	8.2 CU1	**XenServer 8**	
Support for Bromium Secure Platform in Windows VMs	8.2 CU1	8.2 CU1	
Health Check	XenCenter 8.2.6	XenCenter 8.2.7	
Upload of server status reports to Citrix Insight Services (CIS) from XenCenter	XenCenter 8.2.6	XenCenter 8.2.7	Upload your status reports manually through the CIS website.

Item	Deprecation announced in	Removed in	Alternative
PuTTY installed with XenCenter	XenCenter 8.2.6	**XenCenter 2023.3.1**	With future versions of XenCenter, you must install your own instance of PuTTY or OpenSSH on the system where XenCenter is installed.
Localization of XenCenter into Japanese and Simplified Chinese	XenCenter 8.2.6	**XenCenter 2023.3.1**	
Software Fibre Channel over Ethernet (FCoE)	8.2 CU1		
AMD MxGPU	8.2 CU1	**XenServer 8**	
Intel GVT-g	8.2 CU1		
Separate PVS-Accelerator Supplemental Pack	8.2 CU1	**XenServer 8**	These capabilities are now included in the core product.
Support for the following Linux operating systems: CoreOS, Ubuntu 16.04, Debian Jessie 8, Debian Stretch 9, SUSE Linux Enterprise Server 12 SP3, SUSE Linux Enterprise Desktop 12 SP3, CentOS 8	8.2 CU1	8.2 CU1	Upgrade your VMs to a later version of their operating system where available.

Item	Deprecation announced in	Removed in	Alternative
Support for the following Linux operating systems: Ubuntu 18.04, SUSE Linux Enterprise Server 12 SP4, SUSE Linux Enterprise Desktop 12 SP4	8.2 CU1	8.2 CU1	Upgrade your VMs to a later version of their operating system where available.
Support for Windows Server 2012 R2	8.2 CU1	**XenServer 8**	Upgrade your VMs to a later version of their operating system.
Support for Windows Server 2012 and Windows 8.1	8.2 CU1	8.2 CU1	Upgrade your VMs to a later version of their operating system.
Transfer VM	8.2 CU1	8.2 CU1 (XenCenter 8.2.3)	Use the latest release of XenCenter. Since XenCenter 8.2.3, the mechanism used for OVF/OVA import/export and single disk image import has been simplified and these operations are now performed without using the Transfer VM.
Measured Boot Supplemental Pack	8.2 CU1	8.2 CU1	
Container Management Supplemental Pack	8.2	8.2	
Support for Hewlett-Packard Integrated Lights-Out (iLO)	8.2	8.2	

Item	Deprecation announced in	Removed in	Alternative
Support for the following legacy processors: `Xeon E3/5/7 family - Sandy Bridge`, `Xeon E3/5/7 v2 family - Ivy Bridge`	8.2	8.2	
The `guest-tools.iso` file included in the XenServer installation ISO	8.2	8.2	Download the XenServer VM Tools for Windows or for Linux from the XenServer downloads page.
Support for Windows 7, Windows Server 2008 SP2, and Windows Server 2008 R2 SP1	8.2	8.2	Upgrade your VMs to a later version of their operating system.
Legacy SSL mode and support for the TLS 1.0/1.1 protocol	8.2	8.2	
Cross-server private networks	8.2	8.2	
The following xe CLI log commands: `diagnostic-db-log`, `log-set-output`, `log-get-keys`, `log-get`, `log-reopen`	8.2	**XenServer 8**	
The vSwitch Controller (see Notes)	8.1	8.2	

Item	Deprecation announced in	Removed in	Alternative
Legacy partition layouts: DOS partition layout, Old GPT partition layout. Note, this change also removes support for servers with less than 46 GB of primary disk space.	8.1	**XenServer 8**	
VSS and quiesced snapshots	8.1	8.1	
Support for Ubuntu 14.04	8.1	8.1	

Item	Deprecation announced in	Removed in	Alternative
Support for all paravirtualized (PV) VMs, including the following: Red Hat Enterprise Linux 5, Red Hat Enterprise Linux 6, CentOS 5, CentOS 6, Oracle Enterprise Linux 5, Oracle Enterprise Linux 6, Scientific Linux 6, NeoKylin Linux Advanced Server 6.2, Debian Wheezy 7, SUSE Linux Enterprise Server 11 SP3, SUSE Linux Enterprise Server 11 SP4, SUSE Linux Enterprise Server 12, SUSE Linux Enterprise Server 12 SP1, SUSE Linux Enterprise Server 12 SP2, SUSE Linux Enterprise Desktop 11 SP3, SUSE Linux Enterprise Desktop 12, SUSE Linux Enterprise Desktop 12 SP1, SUSE Linux Enterprise Desktop 12 SP2	8.1	8.1	Upgrade your VMs to a later version of their operating system before moving to the latest version of XenServer.
Legacy drivers: `qla4xxx`, `qla3xxx`, `netxen_nic`, `qlge`, `qlcnic`	8.0		

Item	Deprecation announced in	Removed in	Alternative
XenCenter installer bundled with the XenServer installation media.	8.0	8.0	Download the XenCenter installer from the Downloads page instead.
XenCenter connections to XenServer hosts that are version 6.x and earlier.	8.0	8.0	Upgrade your out-of-support XenServer hosts.
Support for Nutanix integration.	8.0	8.0	

Item	Deprecation announced in	Removed in	Alternative
Support for the following legacy processors: `Opteron 13xx Budapest`, `Opteron 23xx/83xx Barcelona`, `Opteron 23xx/83xx Shanghai`, `Opteron 24xx/84xx Istanbul`, `Opteron 41xx Lisbon`, `Opteron 61xx Magny Cours`, `Xeon 53xx Clovertown`, `Xeon 54xx Harpertown`, `Xeon 55xx Nehalem`, `Xeon 56xx Westmere-EP`, `Xeon 65xx/75xx Nehalem-EX`, `Xeon 73xx Tigerton`, `Xeon 74xx Dunnington`	8.0	8.0	For information about supported processors, see the Hardware Compatibility List

Item	Deprecation announced in	Removed in	Alternative
Support for `qemu-trad`. It is no longer possible to use `qemu-trad` by setting `platform-devicemodel=qemu-trad`. All VMs created with `qemu-trad` device profile get automatically upgraded to `qemu-upstream-compat` profile.	8.0	8.0	
Support for the following guest templates: Debian 6 Squeeze, Ubuntu 12.04, Legacy Windows, Asianux Server 4.2, 4.4, and 4.5, NeoKylin Linux Security OS 5, Linx Linux 6, Linx Linux 8, GreatTurbo Enterprise Server 12, Yinhe Kylin 4	8.0	8.0	
Legacy Windows drivers from the Citrix VM Tools ISO	8.0	8.0	

Notes

Backup and restore in xsconsole

The backup and restore capability in xsconsole is temporarily not available. We are actively investigating a replacement solution.

Health Check

Logs for the Health Check service are retained by Windows for troubleshooting purposes. To remove these logs, delete them manually from `%SystemRoot%\System32\Winevt\Logs` on the Windows machine running XenCenter.

Dynamic Memory Control (DMC)

This feature was previously listed as deprecated. The deprecation notice was removed on Jan 30, 2023. DMC is supported in future releases of XenServer.

Disconnecting the vSwitch controller

The vSwitch Controller is no longer supported. Disconnect the vSwitch Controller from your pool before attempting to update or upgrade to the latest version of XenServer.

1. In the vSwitch controller user interface, go to the **Visibility & Control** tab.
2. Locate the pool to disconnect in the **All Resource Pools** table. The pools in the table are listed using the IP address of the pool coordinator.
3. Click the cog icon and select **Remove Pool**.
4. Click **Remove** to confirm.

After the update or upgrade, the following configuration changes take place:

- Cross-server private networks revert to single-server private networks.
- Any Quality of Service settings made through the DVSC console are no longer applied. Network rate limits are no longer enforced.
- ACL rules are removed. All traffic from VMs is allowed.
- Port mirroring (RSPAN) is disabled.

After update or upgrade, if you find any leftover state about the vSwitch Controller in your pool, clear the state with the following CLI command: `xe pool-set-vswitch-controller address=`

Quick start

March 20, 2024

This article steps through how to install and configure XenServer (formerly Citrix Hypervisor) and its graphical, Windows-based user interface, XenCenter. After installation, it takes you through creating

Windows virtual machines (VMs) and then making customized VM templates you can use to create multiple, similar VMs quickly. Finally, this article shows how to create a pool of hosts, which provides the foundation to migrate running VMs between hosts using live migration.

Focusing on the most basic scenarios, this article aims to get you set up quickly.

This article is primarily intended for new users of XenServer and XenCenter. It is intended for those users who want to administer XenServer by using XenCenter. For information on how to administer XenServer using the Linux-based xe
commands through the XenServer command line interface (known as the xe CLI), see Command-line interface.

Terminology and abbreviations

- *Server*: a physical computer that runs XenServer.
- *Host*: a XenServer installation that hosts Virtual Machines (VMs).
- *Virtual Machine (VM)*: 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 NIC.
- *Pool*: a single managed entity that binds together multiple XenServer hosts and their VMs.
- *Pool coordinator* (formerly *pool master*): main host in a pool that provides a single point of contact for all hosts in the pool, routing communication to other members of the pool as necessary.
- *Storage Repository (SR)*: a storage container in which virtual disks are stored.

Major components

XenServer

XenServer is a complete server virtualization platform, with all the capabilities required to create and manage a virtual infrastructure. 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 abstracts elements from the physical machine (such as hard drives, resources, and ports) and allocating them to the virtual machines (VMs) running on it.

XenServer lets you create VMs, take VM disk snapshots, and manage VM workloads.

XenCenter

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

The XenCenter *Online Help* is also a great resource for getting started with XenCenter. Press F1 at any time to access context-sensitive information.

Install XenServer and XenCenter

In this section, you set up a minimum XenServer installation.

What you'll learn

You'll learn how to:

- Install XenServer on a single physical server
- Install XenCenter on a Windows computer
- Connecting XenCenter and XenServer to form the infrastructure for creating and running virtual machines (VMs)

Requirements

To get started, you need the following items:

- A physical computer to be the XenServer host
- A Windows computer to run the XenCenter application
- Installation files for XenServer and XenCenter

The XenServer host computer is dedicated entirely to the task of running XenServer and hosting VMs, and is not used for other applications. The computer that runs XenCenter can be any general-purpose Windows computer that satisfies the hardware requirements. You can use this computer to run other applications too. For more information, see System Requirements.

You can download the installation files from XenServer Downloads.

Install the XenServer host

This is an embedded video. Click the link to watch the video

All servers have at least one IP address associated with them. To configure a static IP address for the host (instead of using DHCP), have the static IP address on hand before beginning this procedure.

> **Tip:**
>
> Press **F12** to advance quickly to the next installer screen. For general help, press **F1**.

To install the XenServer host:

1. Burn the installation files for XenServer to a CD or create a bootable USB.

   > **Note:**
   >
   > For information about using HTTP, FTP, or NFS as your installation source, see Install XenServer.

2. Back up data you want to preserve. Installing XenServer overwrites data on any hard drives that you select to use for the installation.
3. Insert the installation media into the system.
4. Restart the system.
5. Boot from the local installation media (if necessary, see your hardware vendor documentation for information on changing the boot order).
6. Following the initial boot messages and the **Welcome to XenServer** screen, select your keyboard layout for the installation.
7. When the **Welcome to XenServer Setup** screen is displayed, select **Ok**.
8. Read and accept the XenServer EULA.

   > **Note:**
   >
   > If you see a **System Hardware** warning, ensure hardware virtualization assist support is enabled in your system firmware.

9. Select **Ok** to do a clean installation.
10. If you have multiple hard disks, choose a Primary Disk for the installation. Select **Ok**.

    Choose which disks you want to use for virtual machine storage. Choose **Ok**.

11. Select **Local media** as your installation source.
12. Select **Skip Verification**, and then choose **Ok**.

    > **Note:**
    >
    > If you encounter problems during installation, verify the installation source.

13. Create and confirm a root password, which the XenCenter application uses to connect to the XenServer host.

14. Set up the management interface to use to connect to XenCenter.

    If your computer has multiple NICs, select the NIC which you want to use for management traffic (typically the first NIC).

15. Configure the Management NIC IP address with a static IP address or use DHCP.
16. Specify the host name and the DNS configuration manually or automatically through DHCP.

    If you manually configure the DNS, enter the IP addresses of your primary (required), secondary (optional), and tertiary (optional) DNS servers in the fields provided.

17. Select your time zone.
18. Specify how you want the host to determine local time: using NTP or manual time entry. Choose **Ok**.

    - If using NTP, you can specify whether DHCP sets the time server. Alternatively, you can enter at least one NTP host name or IP address in the following fields.
    - If you selected to set the date and time manually, you are prompted to do so.

19. Select **Install XenServer**.

    The installation process starts. This might take some minutes.

20. The next screen asks if you want to install any supplemental packs. Choose **No** to continue.
21. From the **Installation Complete** screen, eject the installation media, and then select **Ok** to reboot the host.

    After the host reboots, XenServer displays **xsconsole**, a system configuration console.

    > **Note:**
    >
    > Make note of the IP address displayed. You use this IP address when you connect XenCenter to the host.

Install XenCenter

XenCenter is typically installed on your local system. You can download the XenCenter installer from the XenServer downloads page

To install XenCenter:

1. Download or transfer the XenCenter installer to the computer that you want to run XenCenter.
2. Double-click the installer `.msi` file to begin the installation.
3. Follow the Setup wizard, which allows you to modify the default destination folder and then to install XenCenter.

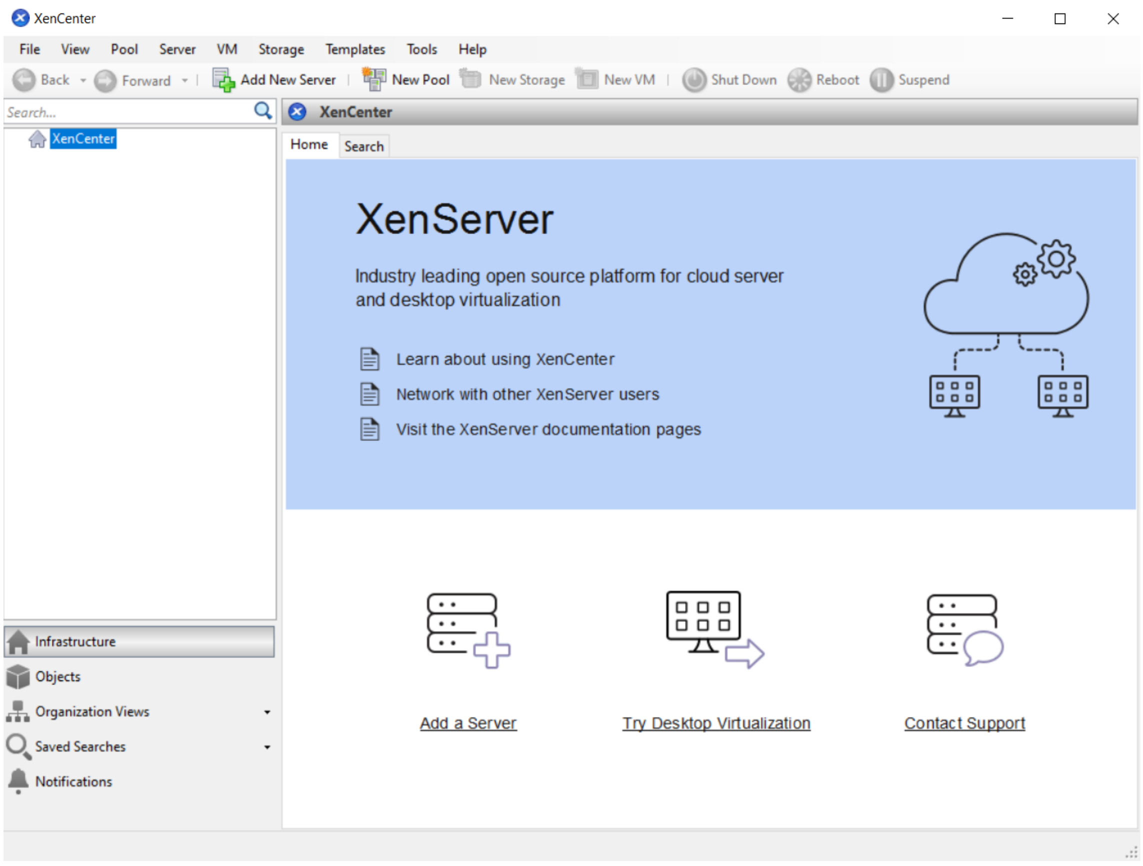
Connect XenCenter to the XenServer host

This procedure enables you to add a host to XenCenter.

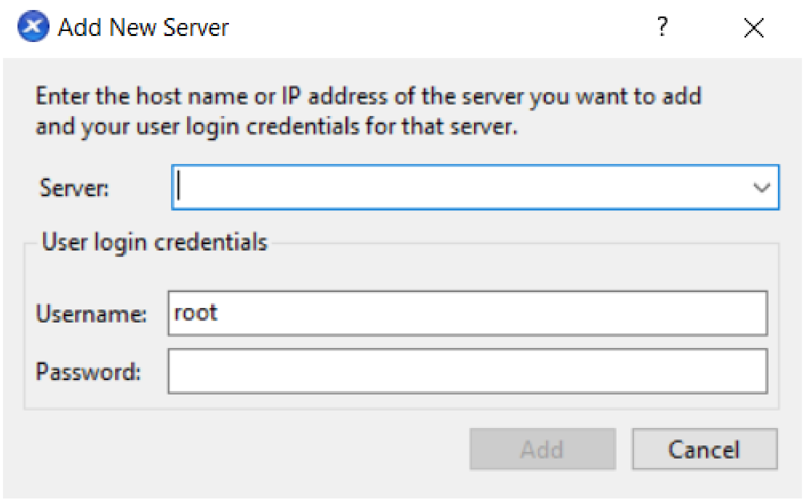
To connect XenCenter to the XenServer host:

1. Launch XenCenter.

   The program opens to the **Home** tab.

   
   XenCenter
File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
Search...
XenCenter
Home Search
XenServer
Industry leading open source platform for cloud server and desktop virtualization
Learn about using XenCenter
Network with other XenServer users
Visit the XenServer documentation pages
Add a Server
Try Desktop Virtualization
Contact Support
Infrastructure
Objects
Organization Views
Saved Searches
Notifications

2. Click the **Add New Server** icon to open the **Add New Server** dialog box.


Add New Server

Enter the host name or IP address of the server you want to add and your user login credentials for that server.

Server:

User login credentials

Username: root

Password:

3. In the **Server** field, enter the IP address of the host. Enter the root user name and password that you set during XenServer installation. Choose **Add**.

> **Note:**
>
> The first time you add a host, the **Save and Restore Connection State** dialog box appears. This dialog box enables you to set your preferences for storing your host connection information and automatically restoring host connections.

License XenServer

You can use XenServer without a license (Trial Edition). The Trial Edition lets you try Premium Edition features, but in a restricted size pool of up to 3 hosts.

If you are using XenServer to run your Citrix Virtual Apps and Desktops workloads, you must have a Premium Edition license. For more information about getting a XenServer license, see https://xenserver.com/buy. Existing Citrix Virtual Apps and Desktops customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

If you have a XenServer license, apply it now. For more information, see Licensing.

Create a pool of XenServer hosts

A resource pool is composed of multiple XenServer host installations, bound together as a single managed entity.

Resource pools enable you to view multiple hosts and their connected shared storage as a single unified resource. You can flexibly deploy VMs across the resource pool based on resource needs and business priorities. A pool can contain up to 64 hosts running the same version of XenServer software, at the same patch level, and with broadly compatible hardware.

One host in the pool is designated as the *pool coordinator*. The pool coordinator provides a single point of contact for the whole pool, routing communication to other members of the pool as necessary. Every member of a resource pool contains all the information necessary to take over the role of pool coordinator if necessary. The pool coordinator is the first host listed for the pool in the XenCenter Resources pane. You can find the pool coordinator's IP address by selecting the pool coordinator and clicking the **Search** tab.

In a pool with shared storage, you can start VMs on *any* pool member that has sufficient memory and dynamically move the VMs between hosts. The VMs are moved while running and with minimal downtime. If an individual XenServer host suffers a hardware failure, you can restart the failed VMs on another host in the same pool.

If the high availability feature is enabled, protected VMs are *automatically* moved if a host fails. On an HA-enabled pool, a new pool coordinator is automatically nominated if the pool coordinator is shut down.

> **Note:**
>
> For a description of heterogeneous pool technology, see Hosts and resource pools.

What you'll learn

You'll learn how to:

- Create a pool of hosts
- Set up a network for the pool
- Bond NICs
- Set up shared storage for the pool

While XenServer accommodates many shared storage solutions, this section focuses on two common types: NFS and iSCSI.

Requirements

To create a pool with shared storage, you need the following items:

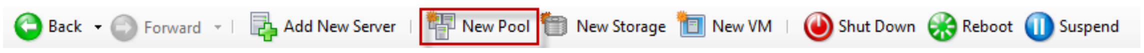
- A second XenServer host, with similar processor type.
  Connect this host to your XenCenter application.
- A storage repository for IP-based storage

To get you started quickly, this section focuses on creating *homogeneous* pools. Within a homogeneous pool, all hosts must have compatible processors and be running the same version of XenServer, under the same type of XenServer product license. For a full list of homogeneous pool requirements, see System requirements.

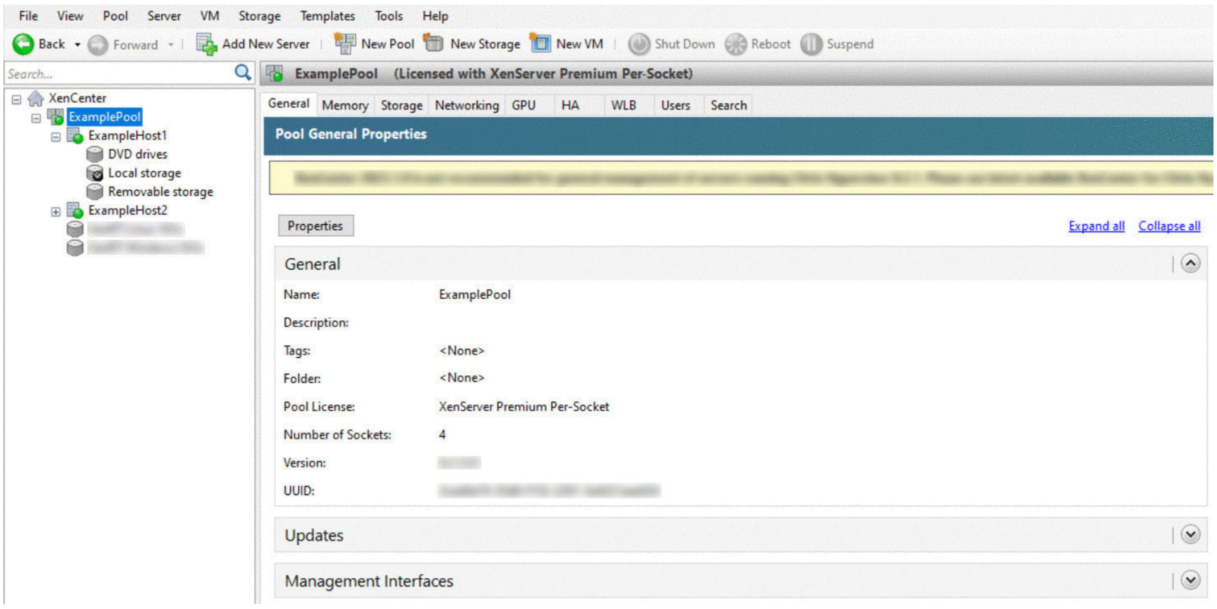
Create a pool

To create a pool:

1. On the toolbar, click the **New Pool** button.

2. Enter a name and optional description for the new pool.
3. Nominate the pool coordinator by selecting a host from the **Coordinator** list.
4. Select the second host to place in the new pool from the **Additional members** list.
5. Click **Create Pool**.
   The new pool appears in the **Resources** pane.


File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
Search...
XenCenter
ExamplePool
ExampleHost1
DVD drives
Local storage
Removable storage
ExampleHost2
General Memory Storage Networking GPU HA WLB Users Search
Properties
Expand all Collapse all

ExamplePool (Licensed with XenServer Premium Per-Socket)

Pool General Properties

[redacted]

General

Name:	ExamplePool
Description:	
Tags:	<None>
Folder:	<None>
Pool License:	XenServer Premium Per-Socket
Number of Sockets:	4
Version:	[redacted]
UUID:	[redacted]

Updates

Management Interfaces

Set up networks for the pool

When you install XenServer, you create a network connection, typically on the first NIC in the pool where you specified an IP address (during XenServer installation).

However, you may need to connect your pool to VLANs and other physical networks. To do so, you must add these networks to the pool. You can configure XenServer to connect each NIC to one physical network and numerous VLANs.

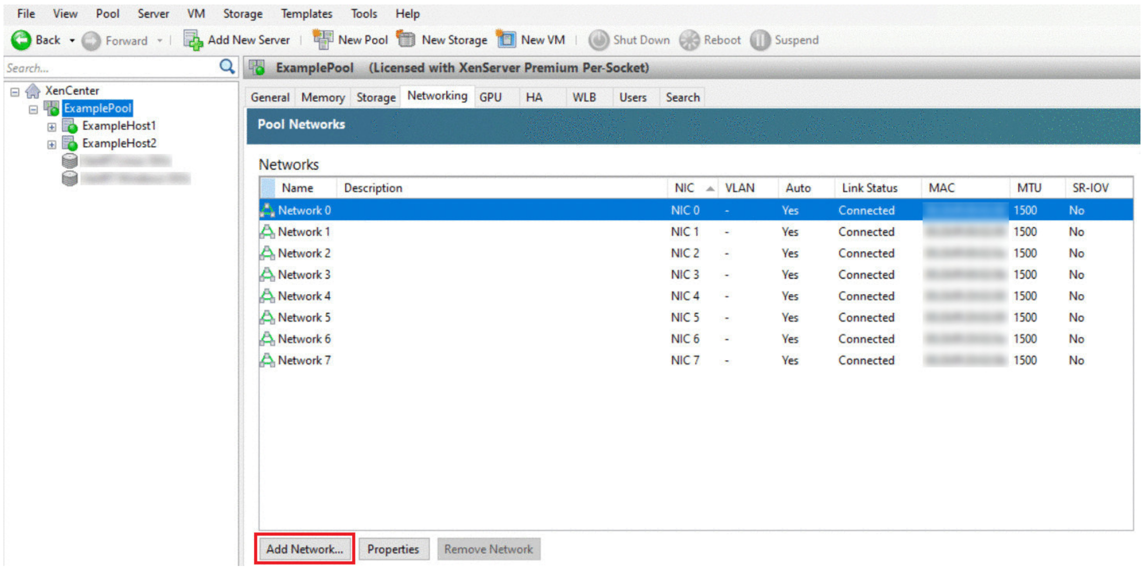
Before creating networks, ensure that the cabling matches on each host in the pool. Plug the NICs on each host into the same physical networks as the corresponding NICs on the other pool members.

> **Note:**
>
> If the NICs were not plugged in to the NICs on the host when you installed XenServer:
>
> - Plug the NICs in
> - In XenCenter, select **<your host> > NICs** tab
> - Click **Rescan** for them to appear

For additional information about configuring XenServer networking, see Networking and About XenServer Networks.

To add a network to XenServer:

1. In the **Resources** pane in XenCenter, select the pool.
2. Click the **Networking** tab.
3. Click **Add Network**.



ExamplePool (Licensed with XenServer Premium Per-Socket)

Networks

Name	Description	NIC	VLAN	Auto	Link Status	MAC	MTU	SR-IOV
Network 0		NIC 0	-	Yes	Connected	[redacted]	1500	No
Network 1		NIC 1	-	Yes	Connected	[redacted]	1500	No
Network 2		NIC 2	-	Yes	Connected	[redacted]	1500	No
Network 3		NIC 3	-	Yes	Connected	[redacted]	1500	No
Network 4		NIC 4	-	Yes	Connected	[redacted]	1500	No
Network 5		NIC 5	-	Yes	Connected	[redacted]	1500	No
Network 6		NIC 6	-	Yes	Connected	[redacted]	1500	No
Network 7		NIC 7	-	Yes	Connected	[redacted]	1500	No

4. On the **Select Type** page, select **External Network**, and click **Next**.
5. On the **Name** page, enter a meaningful name for the network and description.
6. On the **Network settings** page, specify the following:

    - **NIC:** Select the NIC that you want XenServer to use to send and receive data from the network.

- **VLAN:** If the network is a VLAN, enter the VLAN ID (or "tag").
- **MTU:** If the network uses jumbo frames, enter a value for the Maximum Transmission Unit (MTU) between 1500 to 9216. Otherwise, leave the MTU box at its default value of 1500.

If you configure many virtual machines to use this network, you can select the **Automatically add this network to new virtual machines** check box. This option adds the network by default.

7. Click **Finish**.

Bonding NICs

NIC bonding can make your host more resilient by using two or more physical NICs as if they were a single, high-performing channel. This section only provides a very brief overview of bonding, also known as *NIC teaming*. Before configuring bonds for use in a production environment, we recommend reading more in-depth information about bonding. For more information, see Networking.

XenServer supports the following bond modes: Active/active, active/passive (active/backup), and LACP. Active/active provides load balancing and redundancy for VM-based traffic. For other types of traffic (storage and management), active/active cannot load balance traffic. As a result, LACP or multipathing are better choice for storage traffic. For information about multipathing, see Storage. For more information about bonding, see Networking.

LACP options are not visible or available unless you configure the vSwitch as the network stack. Likewise, your switches must support the IEEE 802.3ad standard. The switch must contain a separate LAG group configured for each LACP bond on the host. For more details about creating LAG groups, see Networking.

To bond NICs:

1. Ensure that the NICs you want to bind together are not in use: shut down any VMs with virtual network interfaces using these NICs before creating the bond. After you have created the bond, reconnect the virtual network interfaces to an appropriate network.
2. Select the host in the **Resources** pane then open the **NICs** tab and click **Create Bond**.
3. Select the NICs you want to bond together. To select a NIC, select its check box in the list. Up to four NICs may be selected in this list. Clear the check box to deselect a NIC. To maintain a flexible and secure network, you can bond either two, three, or four NICs when vSwitch is the network stack. However, you can only bond two NICs when Linux bridge is the network stack.
4. Under **Bond** mode, choose the type of bond:
    - Select **Active-active** to configure an active-active bond. Traffic is balanced between the bonded NICs. If one NIC within the bond fails, the host's network traffic automatically routes over the second NIC.

    - Select **Active-passive** to configure an active-passive bond. Traffic passes over only one of the bonded NICs. In this mode, the second NIC only becomes active if the active NIC fails, for example, if it loses network connectivity.
    - Select **LACP with load balancing based on source MAC address** to configure a LACP bond. The outgoing NIC is selected based on MAC address of the VM from which the traffic originated. Use this option to balance traffic in an environment where you have several VMs on the same host. This option is not suitable if there are fewer virtual interfaces (VIFs) than NICs: as load balancing is not optimal because the traffic cannot be split across NICs.
    - Select **LACP with load balancing based on IP and port of source and destination** to configure a LACP bond. The source IP address, source port number, destination IP address, and destination port number are used to allocate the traffic across the NICs. Use this option to balance traffic from VMs in an environment where the number of NICs exceeds the number of VIFs.

      > **Note:**
      >
      > LACP bonding is only available for the vSwitch, whereas active-active and active-passive bonding modes are available for both the vSwitch and Linux bridge. For information about networking stacks, see Networking.

5. To use jumbo frames, set the Maximum Transmission Unit (MTU) to a value between 1500 to 9216.
6. To have the new bonded network automatically added to any new VMs created using the New VM wizard, select the check box.
7. Click **Create** to create the NIC bond and close the dialog box.

   XenCenter automatically moves management and secondary interfaces from secondary bonded NICs to the bond interface when the new bond is created. A host with its management interface on a bond is not permitted to join a pool. Before the host can join a pool, you must reconfigure the management interface and move it back on to a physical NIC.

Setting up shared storage for the pool

To connect the hosts in a pool to a remote storage array, create a XenServer SR. The SR is the storage container where a VM's virtual disks are stored. SRs are persistent, on-disk objects that exist independently of XenServer. SRs can exist on different types of physical storage devices, both internal and external. These types include local disk devices and shared network storage.

You can configure a XenServer SR for various different types of storage, including:

- NFS

- Software iSCSI
- Hardware HBA
- SMB
- Fibre Channel
- Software FCoE (deprecated)

This section steps through setting up two types of shared SRs for a pool of hosts: NFS and iSCSI. Before you create an SR, configure your NFS or iSCSI storage array. Setup differs depending on the type of storage solution that you use. For more information, see your vendor documentation. Generally, before you begin, complete the following setup for your storage solution:

- **iSCSI SR**: You must have created a volume and a LUN on the storage array.
- **NFS SR**: You must have created the volume on the storage device.
- **Hardware HBA**: You must have done the configuration required to expose the LUN before running the New Storage Repository wizard
- **Software FCoE SR**: You must have manually completed the configuration required to expose a LUN to the host. This setup includes configuring the FCoE fabric and allocating LUNs to your SAN's public world wide name (PWWN).

If you are creating an SR for IP-based storage (iSCSI or NFS), you can configure one of the following as the storage network: the NIC that handles the management traffic or a new NIC for the storage traffic. To configure a different a NIC for storage traffic, assign an IP address to a NIC by creating a *management interface*.

When you create a management interface, you must assign it an IP address that meets the following criteria:

- The IP address is on the same subnet as the storage controller, if applicable
- The IP address is on a different subnet than the IP address you specified when you installed XenServer
- The IP address is not on the same subnet as any other management interfaces.

To assign an IP address to a NIC:

1. Ensure that the NIC is on a separate subnet or that routing is configured to suit your network topology. This configuration forces the desired traffic over the selected NIC.
2. In the **Resource** pane of XenCenter, select the pool (or standalone host). Click the **Networking** tab, and then click the **Configure** button.
3. In the **Configure IP Address** dialog, in the left pane, click **Add IP address**.

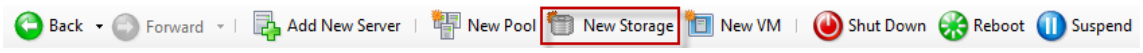
4. Give the new interface a meaningful name (for example, *yourstoragearray_network*). Select the **Network** associated with the NIC that you use for storage traffic.

5. Click **Use these network settings**. Enter a static IP address that you want to configure on the NIC, the subnet mask, and gateway. Click **OK**. The IP address must be on the same subnet as the storage controller the NIC is connected to.

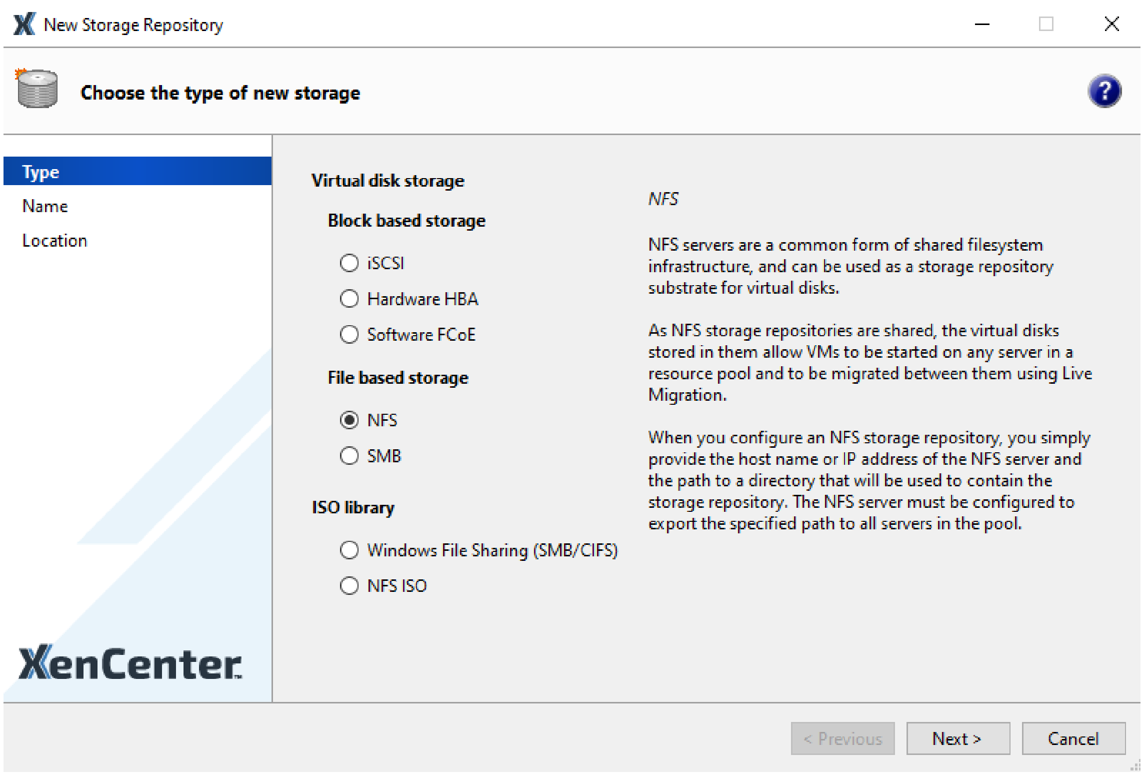
> **Note:**
>
> Whenever you assign a NIC an IP address, it must be on a different subnet than any other NICs with IP addresses in the pool. This includes the primary management interface.

To create a new shared NFS or iSCSI storage repository:

1. On the **Resources** pane, select the pool. On the toolbar, click the **New Storage** button.



The **New Storage Repository** wizard opens.



Selected: NFS

NFS

NFS servers are a common form of shared filesystem infrastructure, and can be used as a storage repository substrate for virtual disks.

As NFS storage repositories are shared, the virtual disks stored in them allow VMs to be started on any server in a resource pool and to be migrated between them using Live Migration.

When you configure an NFS storage repository, you simply provide the host name or IP address of the NFS server and the path to a directory that will be used to contain the storage repository. The NFS server must be configured to export the specified path to all servers in the pool.

2. Under Virtual disk storage, choose NFS or iSCSI as the storage type. Click Next to continue.

3. If you choose NFS:

   a) Enter a name for the new SR and the name of the share where it is located. Click **Scan** to have the wizard scan for existing NFS SRs in the specified location.

> **Note:**
>
> The NFS host must be configured to export the specified path to all XenServer hosts in the pool.

b) Click **Finish**.

   The new SR appears in the **Resources** pane, within the pool.

4. If you choose iSCSI:

   a) Enter a name for the new SR and then the IP address or DNS name of the iSCSI target.

   > **Note:**
   >
   > The iSCSI storage target must be configured to enable every XenServer host in the pool to have access to one or more LUNs.

   b) If you have configured the iSCSI target to use CHAP authentication, enter the user name and password.

   c) Click the **Scan Target Host** button, and then choose the iSCSI target IQN from the Target IQN list.

   > **Warning:**
   >
   > The iSCSI target and all hosts in the pool must have *unique* IQNs.

   d) Click **Target LUN**, and then select the LUN on which to create the SR from the Target LUN list.

   > **Warning:**
   >
   > Each individual iSCSI storage repository must be contained entirely on a single LUN and cannot span more than one LUN. Any data present on the chosen LUN is destroyed.

   e) Click **Finish**.

   The new SR appears in the **Resources** pane, within the pool.

The new shared SR now becomes the default SR for the pool.

Create virtual machines

Through XenCenter, you can create virtual machines in various ways, according to your needs. Whether you are deploying individual VMs with distinct configurations or groups of multiple, similar VMs, XenCenter gets you up and running in just a few steps.

XenServer also provides an easy way to convert batches of virtual machines from VMware. For more information, see Conversion Manager.

This section focuses on a few methods by which to create Windows VMs. To get started quickly, the procedures use the simplest setup of XenServer: a single XenServer host with local storage (after you connect XenCenter to the XenServer host, storage is automatically configured on the local disk of the host).

This section also demonstrates how to use live migration to live migrate VMs between hosts in the pool.

After explaining how to create and customize your new VM, this section demonstrates how to convert that existing VM into a VM template. A VM template preserves your customization so you can always use it to create VMs to the same (or to similar) specifications. It also reduces the time taken to create multiple VMs.

You can also create a VM template from a snapshot of an existing VM. A snapshot is a record of a running VM at a point in time. It saves the storage, configuration, and networking information of the original VM, which makes it useful for backup purposes. Snapshots provide a fast way to make VM templates. This section demonstrates how to take a snapshot of an existing VM and then how to convert that snapshot into a VM template. Finally, this section describes how to create VMs from a VM template.

What you'll learn

You'll learn how to:

- Create a Windows 10 VM
- Install XenServer VM Tools for Windows
- Migrate a running VM between hosts in the pool
- Create a VM template
- Create a VM from a VM template

Requirements

To create a pool with shared storage, you need the following items:

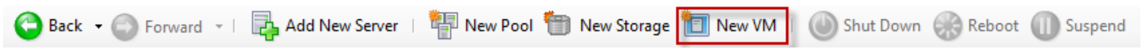
- The XenServer pool you set up
- XenCenter
- Installation files for Windows 10 VM
- Installation files for XenServer VM Tools for Windows

Create a Windows 10 VM

> **Note:**
>
> The following procedure provides an example of creating Windows 10 VM. The default values may vary depending on the operating system that you choose.

To create a Windows VM:

1. On the toolbar, click the **New VM** button to open the New VM wizard.

   The New VM wizard allows you to configure the new VM, adjusting various parameters for CPU, storage, and networking resources.

2. Select a VM template and click **Next**.

   Each template contains the setup information for creating a VM with a specific guest operating system (OS), and with optimum storage. This list reflects the templates that XenServer currently supports.

   > **Note:**
   >
   > If the OS you're installing on your new VM is compatible only with the original hardware, check the **Copy host BIOS strings to VM** box. For example, use this option for an OS installation CD that was packaged with a specific computer.
   >
   > After you first start a VM, you cannot change its BIOS strings. Ensure that the BIOS strings are correct before starting the VM for the first time.

3. Enter a name for and optional description of the new VM.

4. Choose the source of the OS media to install on the new VM.

   Installing from a CD/DVD is the simplest option for getting started. Choose the default installation source option (DVD drive), insert the disk into the DVD drive of the XenServer host, and choose **Next** to proceed.

   XenServer also allows you to pull OS installation media from a range of sources, including a pre-existing ISO library.

   To attach a pre-existing ISO library, click **New ISO library** and indicate the location and type of ISO library. You can then choose the specific operating system ISO media from the list.

5. The VM runs on the installed host. Choose **Next** to proceed.

6. Allocate processor and memory resources.

   Each OS has different configuration requirements which are reflected in the templates. You can choose to modify the defaults if necessary. Click **Next** to continue.

7. Assign a graphics processing unit (GPU).

   The **New VM** wizard prompts you to assign a dedicated GPU or virtual GPUs to the VM. This option enables the VM to use the processing power of the GPU. It provides better support for high-end 3D professional graphics applications such as CAD, GIS, and Medical Imaging applications.

   > **Note:**
   >
   > GPU Virtualization is available for XenServer Premium Edition customers.

8. Configure storage for the new VM.

   Click **Next** to select the default allocation and configuration, or you might want to:

   a) Change the name, description, or size of your virtual disk by clicking **Edit**.

   b) Add a new virtual disk by selecting **Add**.

   > **Note:**
   >
   > When you create a pool of XenServer hosts, you can configure shared storage at this point when creating a VM.

9. Configure networking on the new VM.

   Click **Next** to select the default NIC and configurations, including an automatically created unique MAC address for each NIC, or you can:

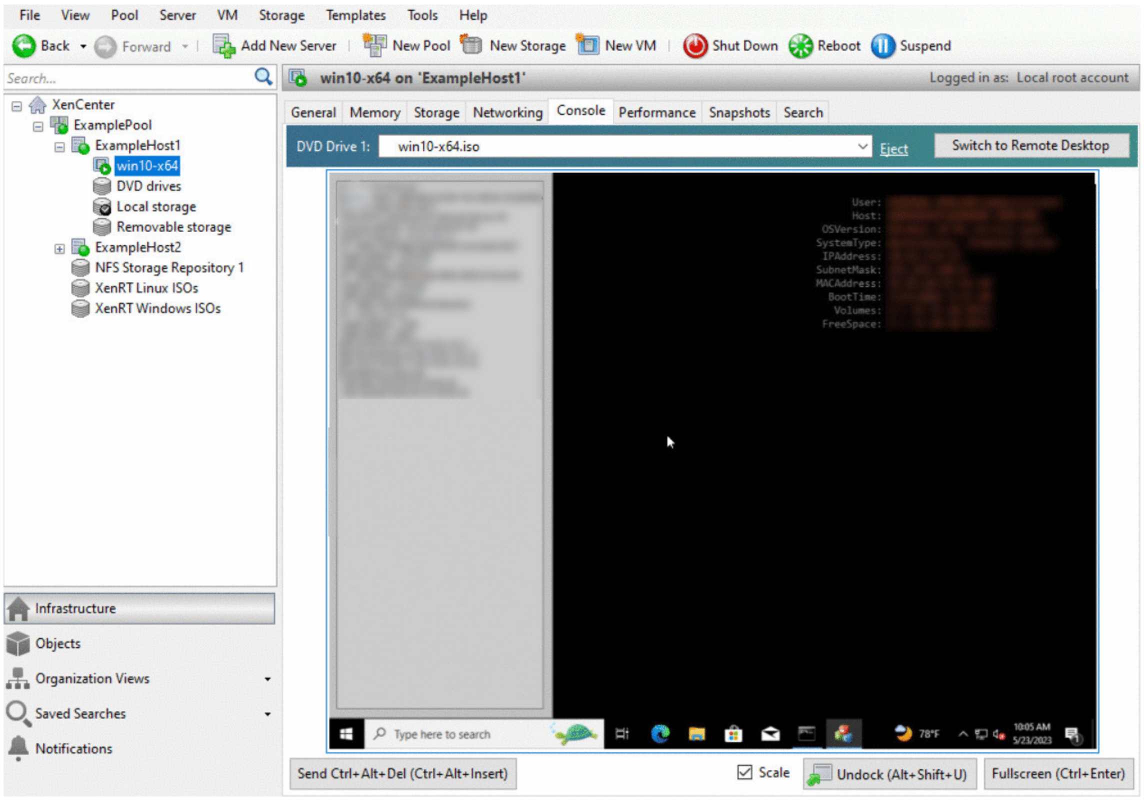
   a) Change the physical network, MAC address, or Quality of Service (QoS) priority of the virtual disk by clicking **Edit**.

   b) Add a new virtual network interface by selecting **Add**.

      XenServer uses the virtual network interface to connect to the physical network on the host. Be sure to select the network that corresponds with the network the virtual machine requires. To add a physical network, see Setting Up Networks for the Pool.

10. Review settings, and then click **Create Now** to create the VM and return to the **Search** tab.

    An icon for your new VM appears under the host in the **Resources** pane.


File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
win10-x64 on 'ExampleHost1'
Logged in as: Local root account
XenCenter
ExamplePool
ExampleHost1
win10-x64
DVD drives
Local storage
Removable storage
ExampleHost2
NFS Storage Repository 1
XenRT Linux ISOs
XenRT Windows ISOs
General Memory Storage Networking Console Performance Snapshots Search
DVD Drive 1: win10-x64.iso Eject
Switch to Remote Desktop
Infrastructure
Objects
Organization Views
Saved Searches
Notifications
Send Ctrl+Alt+Del (Ctrl+Alt+Insert)
Scale
Undock (Alt+Shift+U)
Fullscreen (Ctrl+Enter)

On the **Resources** pane, select the VM, and then click the **Console** tab to see the VM console.

11. Follow the OS installation screens and make your selections.
12. After the OS installation completes and the VM reboots, install the XenServer VM Tools for Windows.

Install XenServer VM Tools for Windows

XenServer VM Tools for Windows provide high performance I/O services without the overhead of traditional device emulation. XenServer VM Tools for Windows consists of I/O drivers (also known as paravirtualized drivers or PV drivers) and the Management Agent. XenServer VM Tools for Windows must be installed on each Windows VM for the VM to have a fully supported configuration. A Windows VM functions without them, but performance is hampered. XenServer VM Tools for Windows also enable certain functions and features, including cleanly shutting down, rebooting, suspending and live migrating VMs.

> **Warning:**
>
> Install XenServer VM Tools for Windows for each Windows VM. Running Windows VMs without XenServer VM Tools for Windows is *not* supported.
>
> We recommend that you snapshot your VM before installing or updating the XenServer VM Tools.

To install XenServer VM Tools for Windows:

1. Download the XenServer VM Tools for Windows file onto your Windows VM. Get this file from the XenServer downloads page.
2. Run the `managementagentx64.msi` file to begin XenServer VM Tools installation.
3. Follow the prompts in the installer.
4. Restart the VM when prompted to complete the installation process.

> **Note:**
>
> I/O drivers are automatically installed on a Windows VM that can receive updates from Windows Update. However, we recommend that you install the XenServer VM Tools for Windows package to install the Management Agent, and to maintain a supported configuration. The following features are available only for XenServer Premium Edition customers:
>
> - Ability to receive I/O drivers from Windows Update
> - Automatic updating of the Management Agent

After you have installed the XenServer VM Tools for Windows, you can customize your VM by installing applications and performing any other configurations. If you want to create multiple VMs with similar specifications, you can do so quickly by making a template from the existing VM. Use that template to create VMs. For more information, see Creating VM Templates.

Migrate running VMs between hosts in a pool

Using live migration, you can move a running VM from one host to another in the same pool, and with virtually no service interruption. Where you decide to migrate a VM to depends on how you configure the VM and pool.

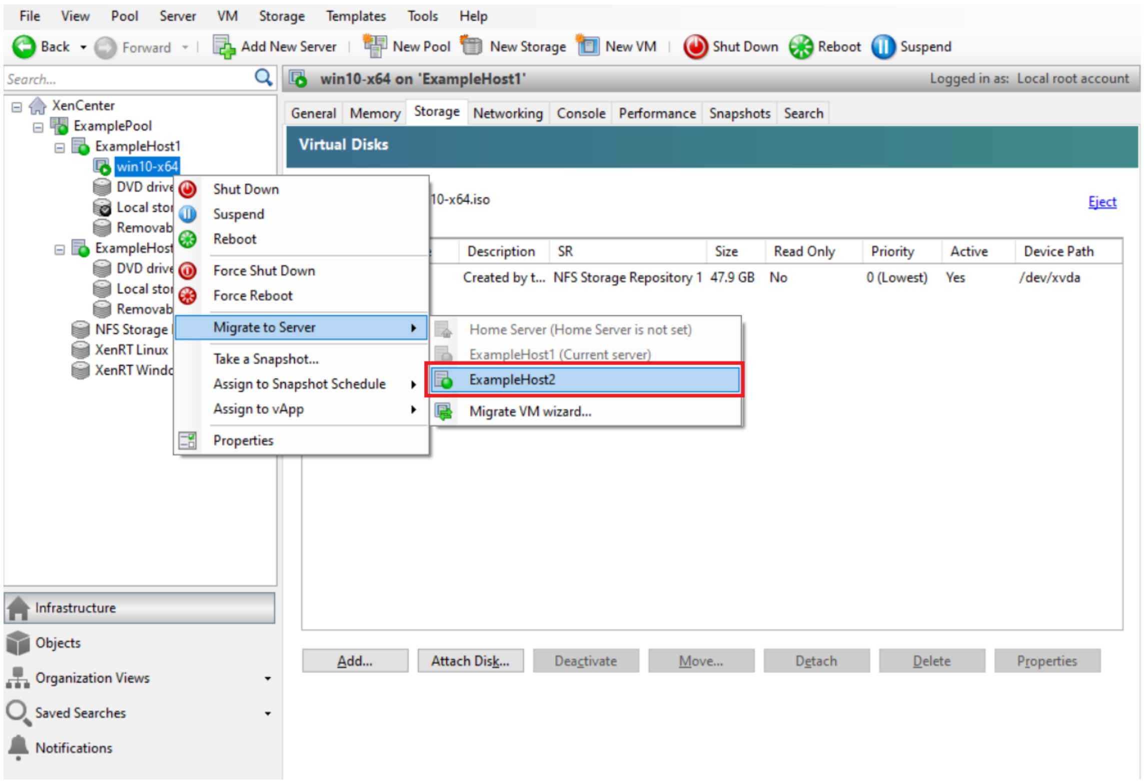
To migrate a running VM:

1. On the **Resources** pane, select the VM that you want to move.

   > **Note:**
   >
   > Ensure that the VM you migrate does not have local storage.

2. Right-click the VM icon, point to **Migrate to Server**, and then select the new VM host.

   > **Tip:**
   >
   > You can also drag the VM onto the target host.



Description	SR	Size	Read Only	Priority	Active	Device Path
Created by t...	NFS Storage Repository 1	47.9 GB	No	0 (Lowest)	Yes	/dev/xvda

3. The migrated VM displays under the new host in the **Resources** pane.

Create VM templates

There are various ways to create a VM template from an existing Windows VM, each with its individual benefits. This section focuses on two methods: converting an existing VM into a template, and creating a template from a snapshot of a VM. In both cases, the VM template preserves the customized configuration of the original VM or VM snapshot. The template can then be used to create new, similar VMs quickly. This section demonstrates how to make new VMs from these templates.

Before you create a template from an existing VM or VM snapshot, we recommend that you run the Windows utility `Sysprep` on the original VM. In general, running `Sysprep` prepares an operating system for disk cloning and restoration. Windows OS installations include many unique elements per installation (including Security Identifiers and computer names). These elements must stay unique and not be copied to new VMs. If copied, confusion and problems are likely to arise. Running `Sysprep` avoids these problems by allowing the generation of new, unique elements for the new VMs.

> **Note:**
>
> Running `Sysprep` may not be as necessary for basic deployments or test environments as it is for production environments.

For more information about `Sysprep`, see your Windows documentation. The detailed procedure of running this utility can differ depending on the version of Windows installed.

Create a VM template from an existing VM To create a VM template from an existing VM:

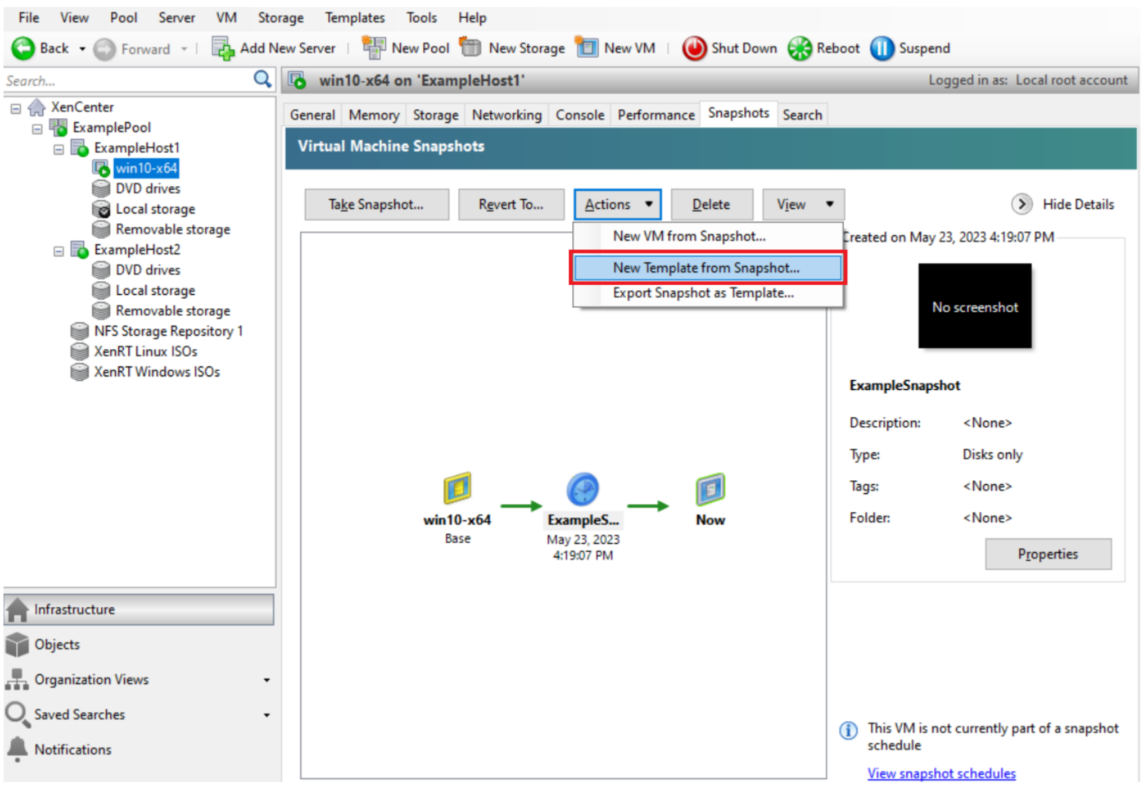
> **Warning:**
>
> When you create a template from an existing VM, the new template replaces the original VM. The VM no longer exists.

1. Shut down the VM that you want to convert.
2. On the **Resources** pane, right-click the VM, and select **Convert to Template**.
3. Click **Convert** to confirm.

   Once you create the template, the new VM template appears in the **Resources** pane, replacing the existing VM.

Create a VM template from a VM snapshot To create a template from a snapshot of a VM:

1. On the **Resources** pane, select the VM. Click the **Snapshots** tab, and then **Take Snapshot**.
2. Enter a name and an optional description of the new snapshot. Click **Take Snapshot**.
3. Once the snapshot finishes and the icon displays in the **Snapshots** tab, select the icon of the new snapshot. From the **Actions** list, choose **New Template from Snapshot**.


File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
win10-x64 on 'ExampleHost1'
Logged in as: Local root account
XenCenter
ExamplePool
ExampleHost1
win10-x64
DVD drives
Local storage
Removable storage
ExampleHost2
DVD drives
Local storage
Removable storage
NFS Storage Repository 1
XenRT Linux ISOs
XenRT Windows ISOs
General Memory Storage Networking Console Performance Snapshots Search
Virtual Machine Snapshots
Take Snapshot... Revert To... Actions Delete View
Hide Details
New VM from Snapshot...
New Template from Snapshot...
Export Snapshot as Template...
Created on May 23, 2023 4:19:07 PM
No screenshot
ExampleSnapshot
Description: <None>
Type: Disks only
Tags: <None>
Folder: <None>
Properties
win10-x64
Base
ExampleS...
May 23, 2023
4:19:07 PM
Now
This VM is not currently part of a snapshot schedule
View snapshot schedules
Infrastructure
Objects
Organization Views
Saved Searches
Notifications

4. Enter a name for the template, and then click **Create**.

Create VMs from a VM template

To create a VM from a customized VM template:

1. On the XenCenter **Resources** pane, right-click the template, and select **New VM** wizard.

   The **New VM** wizard opens.

2. Follow the **New VM** wizard to create a VM from the selected template.

   > **Note:**
   >
   > When the wizard prompts you for an OS installation media source, select the default and continue.

   The new VM appears in the **Resources** pane.

If you are using a template created from an existing VM, you can also choose to select **Quick Create**. This option does not take you through the **New VM** wizard. Instead this option instantly creates and provisions a new VM using all the configuration settings specified in your template.

System requirements

February 1, 2024

XenServer requires at least two separate physical x86 computers: one to be the XenServer host and the other to run the XenCenter application or the XenServer Command-Line Interface (CLI). The XenServer host computer is dedicated entirely to the task of running XenServer and hosting VMs, and is not used for other applications.

> **Warning:**
>
> XenServer supports only drivers and supplemental packs that are provided by us being installed directly in the host's control domain. Drivers provided by third-party websites, including drivers with the same name or version number as those we provide, are not supported.
>
> The following exceptions are supported:
>
> - Software supplied as a supplemental pack and explicitly endorsed by us.
> - Drivers that NVIDIA provides to enable vGPU support. For more information, see NVIDIA vGPU.

> Other drivers provided by NVIDIA, for example, the Mellanox drivers, are not supported with XenServer unless distributed by us.

To run XenCenter use any general-purpose Windows system that satisfies the hardware requirements. This Windows system can be used to run other applications.

When you install XenCenter on this system, the XenServer CLI is also installed. A standalone remote XenServer CLI can be installed on any RPM-based Linux distribution. For more information, see Command-line interface.

XenServer host system requirements

Although XenServer is usually deployed on server-class hardware, XenServer is also compatible with many models of workstations and laptops. For more information, see the Hardware Compatibility List (HCL).

The following section describes the recommended XenServer hardware specifications.

The XenServer host must be a 64-bit x86 server-class machine devoted to hosting VMs. XenServer creates an optimized and hardened Linux partition with a Xen-enabled kernel. This kernel controls the interaction between the virtualized devices seen by VMs and the physical hardware.

XenServer can use:

- Up to 6 TB of RAM
- Up to 16 physical NICs
- Up to 448 logical processors per host.

  > **Note:**
  >
  > The maximum number of logical processors supported differs by CPU. For more information, see the Hardware Compatibility List (HCL).

The system requirements for the XenServer host are:

CPUs

One or more 64-bit x86 CPUs, 1.5 GHz minimum, 2 GHz or faster multicore CPU recommended.

To support VMs running Windows or more recent versions of Linux, you require an Intel VT or AMD-V 64-bit x86-based system with one or more CPUs.

> **Note:**
>
> Ensure that you enable hardware support for virtualization on the XenServer host. Virtualization support is an option in your system firmware. It is possible that your hardware might have virtualization support disabled. For more information, see your server documentation.

To support VMs running supported paravirtualized Linux, you require a standard 64-bit x86-based system with one or more CPUs.

RAM

2 GB minimum, 4 GB or more recommended

Disk space

- Locally attached storage with 46 GB of disk space minimum, 70 GB of disk space recommended
- SAN via HBA (not through software) when installing with multipath boot from SAN.

For a detailed list of compatible storage solutions, see the Hardware Compatibility List (HCL).

Network

100 Mbit/s or faster NIC. One or more Gb, or 10 Gb NICs is recommended for faster export/import data transfers and VM live migration.

We recommend that you use multiple NICs for redundancy. The configuration of NICs differs depending on the storage type. For more information, see the vendor documentation.

XenServer requires an IPv4 network for management and storage traffic.

> **Notes:**
>
> - Ensure that the time setting on your server is set to the current time in UTC.
> - In some support cases, serial console access is required for debug purposes. When setting up the XenServer configuration, we recommend that you configure serial console access. For hosts that do not have physical serial port or where suitable physical infrastructure is not available, investigate whether you can configure an embedded management device. For example, Dell DRAC. For more information about setting up serial console access, see CTX228930 - How to Configure Serial Console Access on XenServer 7.0 and later.

XenCenter system requirements

XenCenter has the following system requirements:

- **Operating System:**
  - Windows 10
  - Windows 11
  - Windows Server 2016
  - Windows Server 2019
- **.NET Framework:** Version 4.8
- **CPU Speed:** 750 MHz minimum, 1 GHz or faster recommended
- **RAM:** 1 GB minimum, 2 GB or more recommended
- **Disk Space:** 100 MB minimum
- **Network:** 100 Mbit/s or faster NIC
- **Screen Resolution:** 1024x768 pixels, minimum

If you want XenCenter to be able to launch an external SSH console that connects to your server, install one of the following applications on the system:

- PuTTY
- OpenSSH (installed by default on some Windows operating systems)

For more information, see Configure XenCenter to use an external SSH console.

Supported guest operating systems

For a list of supported VM operating systems, see Guest operating system support.

Pool requirements

A resource pool is a homogeneous or heterogeneous aggregate of one or more hosts, up to a maximum of 64. Before you create a pool or join a host to an existing pool, ensure that all hosts in the pool meet the following requirements.

Hardware requirements

All of the servers in a XenServer resource pool must have broadly compatible CPUs, that is:

- The CPU vendor (Intel, AMD) must be the same on all CPUs on all servers.
- All CPUs must have virtualization enabled.

Other requirements

In addition to the hardware prerequisites identified previously, there are some other configuration prerequisites for a host joining a pool:

- It must have a consistent IP address (a static IP address on the host or a static DHCP lease). This requirement also applies to the servers providing shared NFS or iSCSI storage.
- Its system clock must be synchronized to the pool coordinator (for example, through NTP).
- It cannot be a member of an existing resource pool.
- It cannot have any running or suspended VMs or any active operations in progress on its VMs, such as shutting down or exporting. Shut down all VMs on the host before adding it to a pool.
- It cannot have any shared storage already configured.
- It cannot have a bonded management interface. Reconfigure the management interface and move it on to a physical NIC before adding the host to the pool. After the host has joined the pool, you can reconfigure the management interface again.
- It must be running the same version of XenServer, at the same patch level, as hosts already in the pool.
- It must be configured with the same supplemental packs as the hosts already in the pool. Supplemental packs are used to install add-on software into the XenServer control domain, dom0. To prevent an inconsistent user experience across a pool, all hosts in the pool must have the same supplemental packs at the same revision installed.
- It must have the same XenServer license as the hosts already in the pool. You can change the license of any pool members after joining the pool. The host with the lowest license determines the features available to all members in the pool.

XenServer hosts in resource pools can contain different numbers of physical network interfaces and have local storage repositories of varying size.

> **Note:**
>
> Servers providing shared NFS or iSCSI storage for the pool must have a static IP address or be DNS addressable.

Homogeneous pools

A homogeneous resource pool is an aggregate of servers with identical CPUs. CPUs on a server joining a homogeneous resource pool must have the same vendor, model, and features as the CPUs on servers already in the pool.

Heterogeneous pools

Heterogeneous pool creation is made possible by using technologies in 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 feature set than it actually does. These capabilities enable you to create pools of hosts with different CPUs but still safely support live migrations.

For information about creating heterogeneous pools, see Hosts and resource pools.

Configuration limits

April 24, 2024

Use the following configuration limits as a guideline when selecting and configuring your virtual and physical environment for XenServer. The following tested and recommended configuration limits are fully supported for XenServer.

- Virtual machine limits
- XenServer host limits
- Resource pool limits

Factors such as hardware and environment can affect the limitations listed below. More information about supported hardware can be found on the Hardware Compatibility List. Consult your hardware manufacturers'documented limits to ensure that you do not exceed the supported configuration limits for your environment.

Virtual machine (VM) limits

Item	Limit
Compute	
Virtual CPUs per VM (Linux)	32/64 (see note 1)
Virtual CPUs per VM (Windows)	32/64 (see note 1)
Memory	
RAM per VM	1.5 TiB (see note 2)

Item	Limit
Storage	
Virtual Disk Images (VDI) (including CD-ROM) per VM	241 (see note 3)
Virtual CD-ROM drives per VM	1
Virtual Disk Size (NFS)	2040 GiB
Virtual Disk Size (LVM)	2040 GiB
Virtual Disk Size (GFS2)	16 TiB
Networking	
Virtual NICs per VM	7 (see note 4)
Graphics Capability	
vGPUs per VM	8
Passed through GPUs per VM	1
Devices	
Pass-through USB devices	6

Notes:

1. Consult your guest OS documentation to ensure that you do not exceed the supported limits. As of yet, Red Hat Enterprise Linux 8 and derivatives do not support more than 32 vCPUs. Though the limit is 64, we recommend to set the limit to 32 if your VMs might not be trustworthy or if you want to prevent a potential impact on system availability.
2. The maximum amount of physical memory addressable by your operating system varies. Setting the memory to a level greater than the operating system supported limit may lead to performance issues within your guest.
3. The maximum number of VDIs supported depends on the guest operating system. Consult your guest operating system documentation to ensure that you do not exceed the supported limits.

> 4. Several guest operating systems have a lower limit, other guests require installation of the XenServer VM Tools to achieve this limit.

XenServer host limits

Item	Limit
Compute	
Logical processors per host	960 (see note 1)
Concurrent VMs per host	1000 (see note 2)
Concurrent protected VMs per host with HA enabled	500
Virtual GPU VMs per host	128 (see note 3)
Memory	
RAM per host	6 TB
Storage	
Concurrent active virtual disks per host	2048 (see note 4)
Storage repositories per host (NFS)	400
Networking	
Physical NICs per host	16
Physical NICs per network bond	4
Virtual NICs per host	512
VLANs per host	800
Network Bonds per host	4
Graphics Capability	
GPUs per host	8 (see note 5)

Notes:

1. The maximum number of logical physical processors supported differs by CPU. For more information, see the Hardware Compatibility List.
2. The maximum number of VMs per host supported depends on VM workload, system load, network configuration, and certain environmental factors. We reserve the right to determine what specific environmental factors affect the maximum limit at which a system can function. For larger pools (over 32 hosts), we recommend allocating at least 8GB RAM to the Control Domain (Dom0). For systems running over 500 VMs or when using the PVS Accelerator, we recommend allocating at least 16 GB RAM to the Control Domain. For information about configuring Dom0 memory, see CTX134951 - How to Configure dom0 Memory.
3. For NVIDIA vGPU, 128 vGPU accelerated VMs per host with 4xM60 cards (4x32=128 VMs), or 2xM10 cards (2x64=128 VMs). For Intel GVT-g, 7 VMs per host with a 1,024 MB aperture size. Smaller aperture sizes can further restrict the number of GVT-g VMs supported per host. This figure might change. For the current supported limits, see the Hardware Compatibility List.
4. The number of concurrent active virtual disks per host is also constrained by the number of SRs you have attached to the host and the number of attached VDIs that are allowed for each SR (600). For more information, see the "Attached VDIs per SR"entry in the Resource pool limits.
5. This figure might change. For the current supported limits, see the Hardware Compatibility List.

Resource pool limits

Item	Limit
Compute	
VMs per resource pool	2400
Hosts per resource pool	64 (see note 1)
Networking	
VLANs per resource pool	800
Disaster recovery	

Item	Limit
Integrated site recovery storage repositories per resource pool	8
Storage	
Paths to a LUN	16
Multipathed LUNs per host	150 (see note 2)
Multipathed LUNs per host (used by storage repositories)	150 (see note 2)
VDIs per SR (NFS, SMB, EXT, XFS, GFS2)	20000
VDIs per SR (LVM)	1000
Attached VDIs per SR (all types)	600
Storage repositories per pool (NFS)	400
Storage repositories per pool (GFS2)	62
Maximum file system size (GFS2)	100 TiB
Storage live migration	
(non-CDROM) VDIs per VM	6
Snapshots per VM	1
Concurrent transfers	3
XenCenter	
Concurrent operations per pool	25

Notes:

1. Clustered pools that use GFS2 storage support a maximum of 16 hosts in the resource pool.
2. When HA is enabled, we recommend increasing the default timeout to at least 120 seconds when more than 30 multipathed LUNs are present on a host. For information about increasing the HA timeout, see Configure high availability timeout.

Hardware drivers

March 28, 2024

We collaborate with partner organizations to provide drivers and support for a wide range of hardware. For more information, see the Hardware Compatibility List.

To support this hardware, your installation of XenServer 8 includes third-party drivers that have been certified as compatible with XenServer. A list of the drivers included in-box with your initial XenServer installation is given in the summary article Driver versions for XenServer and Citrix Hypervisor.

Updates to drivers

We regularly deliver updated versions of these drivers which can enable new hardware or resolve issues with existing hardware. Most driver updates are delivered through the updates mechanism. For more information, see Update your XenServer hosts.

Some driver updates are released as driver disk ISO files on the website https://support.citrix.com. These drivers are listed in the summary article Driver versions for XenServer and Citrix Hypervisor.

Even though we distribute the drivers and their source code for our customers to use, the hardware vendor owns the driver source files.

Support for drivers

XenServer supports only drivers that are delivered in-box with the product or are downloaded from https://support.citrix.com. Drivers provided by third-party websites, including drivers with the same name or version number as those drivers provided by us, are not supported.

> **Note:**
>
> The only exception to this restriction are the drivers that NVIDIA provides to enable vGPU support. For more information, see NVIDIA vGPU.
>
> Other drivers provided by NVIDIA, for example, the Mellanox drivers, are only supported with XenServer when distributed by us.

Do not download drivers from your hardware vendor website, even if the driver has the same version number as the one provided by XenServer. These drivers are not supported.

Before a driver can be supported with XenServer, it must be certified with us and released through one of the approved mechanisms. This certification process ensures that the driver is of a format required to be installable in a XenServer environment and that it is compatible with XenServer 8.

What if a driver I need isn't supported?

If your hardware vendor recommends that you install a specific driver version that is not available in-box or on the https://support.citrix.com website, request that the vendor contacts us to certify this version of the driver with XenServer.

We provide the vendors with certification kits that they can use to test updated versions of their drivers that are required by the shared customer base of Citrix Hypervisor and the hardware vendor. After the vendor provides us with the certification test results, we validate that those results show no issues or regressions in the updated version of the driver. The driver version is now certified with XenServer and we publish the driver our regular updates or as a driver disk ISO on https://support.citrix.com.

For more information about the certification process the vendor must follow, see the article Hardware Compatibility List explained.

Guest operating system support

March 21, 2024

When installing VMs and allocating resources such as memory and disk space, follow the guidelines of the operating system and any relevant applications.

Operating System	Minimum RAM	Maximum RAM	Minimum Disk Space
Windows 10 (64-bit)	2 GB	1.5 TB	32 GB (40 GB or more recommended)
Windows 11 (64-bit)	4 GB	1.5 TB	32 GB (40 GB or more recommended)
Windows Server 2016, Windows Server Core 2016 (64-bit)	1 GB	1.5 TB	32 GB (40 GB or more recommended)
Windows Server 2019, Windows Server Core 2019 (64-bit)	1 GB	1.5 TB	32 GB (40 GB or more recommended)
Windows Server 2022, Windows Server Core 2022 (64-bit)	1 GB	1.5 TB	32 GB (40 GB or more recommended)
CentOS 7 (64-bit)	2 GB	1.5 TB	10 GB

Operating System	Minimum RAM	Maximum RAM	Minimum Disk Space
CentOS Stream 9 (64-bit) (preview) (see note 1)	2 GB	1.5 TB	10 GB
Red Hat Enterprise Linux 7 (64-bit)	2 GB	1.5 TB	10 GB
Red Hat Enterprise Linux 8 (64-bit)	2 GB	1.5 TB	10 GB
Red Hat Enterprise Linux 9 (64-bit) (preview) (see note 1)	2 GB	1.5 TB	10 GB
SUSE Linux Enterprise Server 12 SP5 (64-bit)	1 GB	1.5 TB	8 GB
SUSE Linux Enterprise Server 15 SP1, 15 SP2, 15 SP3, 15 SP4, 15 SP5 (64-bit)	1 GB	1.5 TB	8 GB
SUSE Linux Enterprise Desktop 15 SP4, 15 SP5 (64-bit)	1 GB	1.5 TB	8 GB
Oracle Linux 7 (64-bit)	2 GB	1.5 TB	10 GB
Oracle Linux 8 (64-bit)	2 GB	1.5 TB	10 GB
Scientific Linux 7 (64-bit)	2 GB	1.5 TB	10 GB
Debian Buster 10 (64-bit)	512 MB	1.5 TB	10 GB
Debian Bullseye 11 (64-bit)	512 MB	1.5 TB	10 GB
Debian Bookworm 12 (64-bit) (preview) (see note 1)	1 GB	1.5 TB	10 GB
Ubuntu 20.04 (64-bit)	512 MB	1.5 TB	10 GB
Ubuntu 22.04 (64-bit)	1 GB	1.5 TB	10 GB
NeoKylin Linux Advanced Server 7.2 (64-bit)	1 GB	1.5 TB	10 GB
Gooroom 2 (64-bit)	1 GB	1.5 TB	10 GB

Operating System	Minimum RAM	Maximum RAM	Minimum Disk Space
Rocky Linux 8 (64-bit)	1 GB	1.5 TB	10 GB
Rocky Linux 9 (64-bit) (preview) (see note 1)	2 GB	1.5 TB	15 GB

> **Note:**
>
> 1. Customers who wish to use this guest OS must also install XenServer VM Tools for Linux v8.3.1-1 or later, available to download from the XenServer product downloads page.

- XenServer VM Tools for Linux is only supported on the Linux guest operating systems listed above.
- All supported operating systems run in HVM mode.
- Individual versions of the operating systems can also impose their own maximum limits on the amount of memory supported (for example, for licensing reasons).
- When configuring guest memory, do not to exceed the maximum amount of physical memory that your operating system can address. Setting a memory maximum that is greater than the operating system supported limit might lead to stability problems within your guest.
- To create a VM of a newer minor version of RHEL than is listed in the preceding table, use the following method:
  - Install the VM from the latest supported media for the major version
  - Use `yum update` to update the VM to the newer minor version

  This approach also applies to RHEL-based operating systems such as CentOS and Oracle Linux.
- XenServer supports all SKUs (editions) for the listed versions of Windows.

Long-term guest support

XenServer includes a long-term guest support (LTS) policy for Linux VMs. The LTS policy enables you to consume minor version updates by one of the following methods:

- Installing from new guest media
- Upgrading from an existing supported guest

Out-of-support operating systems

The list of supported guest operating systems can contain operating systems that were supported by their vendors at the time this version of XenServer was released, but are now no longer supported by their vendors.

We no longer offer support for these operating systems (even if they remain listed in the table of supported guests or their templates remain available on your XenServer hosts). While attempting to address and resolve a reported issue, we assess if the issue directly relates to an out-of-support operating system on a VM. To assist in making that determination, we might ask you to attempt to reproduce an issue using a supported version of the guest operating system. If the issue seems to be related to the out-of-support operating system, we will not investigate the issue further.

> **Note:**
>
> Windows versions that are supported by Microsoft as part of an LTSB branch are supported by XenServer.
> Windows versions that are out of support, but part of an Extended Security Updates (ESU) agreement are not supported by XenServer.

Connectivity requirements

April 8, 2024

This article provides an overview of domains and common ports that are used by XenServer components and must be considered as part of the networking architecture, especially if communication traffic traverses network components such as firewalls or proxy servers where ports must be opened or domains added to an allow list to ensure communication flow.

External domains accessed by XenServer product components

Depending on your deployment and requirements, configure your firewall to enable these XenServer components to access the listed domains.

XenServer hosts

Your XenServer hosts access the following domains:

Domain	Port	Direction	Details
`repo.ops.xenserver.com`	443	Outbound	The XenServer pool coordinator downloads available updates for XenServer 8 from this location. For more information, see Updates.
`repo-src.ops.xenserver.com`	443	Outbound	The XenServer pool coordinator downloads the source files for XenServer 8 updates from this location. For more information, see Updates.
`telemetry.ops.xenserver.com`	443	Outbound	The XenServer pool coordinator gathers telemetry data and uploads it regularly to this location. For more information, see Telemetry.

When configuring your XenServer pools to receive updates, you can configure a proxy server for the pool coordinator to use to download the updates. For more information, see Configure updates for your pool.

XenCenter

The XenCenter management console accesses the following domains:

Domain	Port	Direction	Details
`updates.ops.xenserver.com`	443	Outbound	XenCenter polls information on this site to see whether updates are available for XenCenter and for XenServer 8 hosts. For more information, see Update your XenServer hosts
`citrix.com` and subdomains	443	Outbound	If you use XenCenter to administer Citrix Hypervisor 8.2 Cumulative Update 1 hosts and pool, XenCenter accesses subdomains on the `citrix.com` domain to be notified about and to download hotfixes. For more information, see Update your Citrix Hypervisor hosts

You can configure a proxy server that XenCenter goes through to check for and download updates. For more information, see Proxy server.

Windows VMs

If you have set up your Windows VMs to receive updates to the XenServer VM Tools management agent, your Windows VM accesses the following domains:

Domain	Port	Direction	Details
`pvupdates.vmd.citrix.com`	443	Outbound	The XenServer VM Tools for Windows poll information on this site to see whether updates are available for the management agent.
`downloadns.citrix.com.edgesuite.net`	443	Outbound	The XenServer VM Tools for Windows download the installer files for the management agent from this location.

If you don't want your Windows VM to access these domains, you can redirect management agent updates to an internal web server. For more information, see Redirect the Management Agent updates.

Communication ports used by XenServer product components

The ports listed in the following table are the common ports that are used by XenServer components. Not all ports need to be open, depending on your deployment and requirements.

Source	Destination	Type	Port	Details
XenServer hosts	XenServer hosts	TCP	80, 443	Intra-host communication between members of a resource pool using the management API
	Citrix License Server	TCP	27000	Handles initial connection for license requests
		TCP	7279	Check-in/check-out of licenses

Source	Destination	Type	Port	Details
	NTP Service	TCP, UDP	123	Time Synchronization
	DNS Service	TCP, UDP	53	DNS Lookups
	Domain Controller	TCP, UDP	389	LDAP (for Active Directory user authentication)
		TCP	636	LDAP over SSL (LDAPS)
	FileServer (with SMB storage)	TCP, UDP	139	ISOStore:NetBIOSSessionServi
		TCP, UDP	445	ISOStore:Microsoft-DS
	SAN Controller	TCP	3260	iSCSI Storage
	NAS Head/File Server	TCP	2049	NFSv4 Storage
		TCP, UDP	2049	NFSv3 Storage. TCP is the default
		TCP, UDP	111	NFSv3 Storage - connection to rpcbind
		TCP, UDP	Dynamic	NFSv3 Storage - a dynamic set of ports chosen by the filer
	Syslog	UDP	514	Sends data to a central location for collation
	Clustering	TCP	8892, 8896, 21064	Communication between all pool members in a clustered pool
		UDP	5404, 5405	
XenCenter	XenServer hosts	TCP	22	SSH
		TCP	443	Management using the management API

Source	Destination	Type	Port	Details
	Virtual Machine	TCP	5900	VNC for Linux VMs
		TCP	3389	RDP for Windows VMs
Workload Balancing virtual appliance	XenServer hosts	TCP	8012	By default, the Workload Balancing server uses 8012. However, if you specify a different port during Workload Balancing set up, ensure that communication is allowed on that port.
Other clients	XenServer hosts	TCP	80, 443	Any client that uses the management API to communicate with XenServer hosts

XenServer interoperates with various Citrix products. For more information about the ports these products use, see Communication ports used by Citrix.

> **Note:**
>
> - To improve security, you can close TCP port 80 on the management interface of XenServer hosts. For more information about how to close port 80, see Restrict use of port 80.
> - If FQDN is used instead of IP as a resource, then make sure it is resolvable.

Active Directory integration

If you use Active Directory in your environment, ensure that the following firewall ports are open for outbound traffic 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
636	UDP/TCP	LDAP over SSL
3268	TCP	Global Catalog Search

For more information, see Active Directory integration

Citrix Provisioning Services

If you use Citrix Provisioning Services in your environment, ensure that the following firewall ports can be accessed:

Port	Protocol	Use
6901, 6902, 6905	UDP	Provisioning server outbound communication (packets destined for the target device)
6910	UDP	Target device logon with Citrix Provisioning Services
6901	UDP	Configurable target device port. The default port is 6901.
6910–6930	UDP	Configurable server port range. The default range is 6910–6930.

For more information, see Citrix Provisioning Services and Communication ports used by Citrix.

Technical overview

March 14, 2024

XenServer (formerly Citrix Hypervisor) is an industry leading platform for cost-effective desktop, server, and cloud virtualization infrastructures. XenServer enables organizations of any size or type to consolidate and transform compute resources into virtual workloads for today's data center requirements. Meanwhile, it ensures a seamless pathway for moving workloads to the cloud.

The key features of XenServer are:

- Consolidating multiple virtual machines (VMs) onto a physical server
- Reducing the number of separate disk images to be managed
- Allowing for easy integration with existing networking and storage infrastructures
- Enabling you to schedule zero downtime maintenance by live migrating VMs between XenServer hosts
- Assuring availability of VMs by using high availability to configure policies that restart VMs on another host in case one fails
- Increasing portability of VM images, as one VM image works on a range of deployment infrastructures

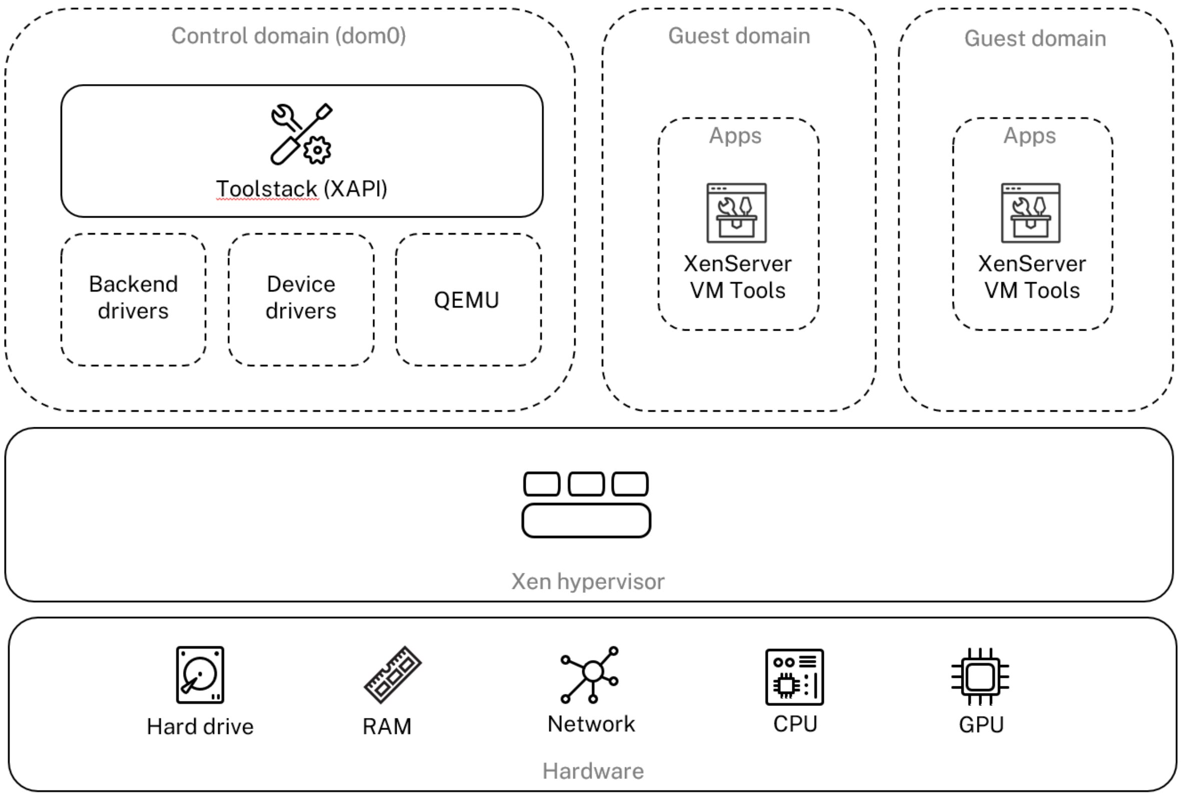
Virtualization and hypervisor

Virtualization, or to be more specific, hardware virtualization, is a method of running multiple independent VMs on a single physical computer. Software run on these virtual machines is separated from the underlying hardware resources. It's a way of fully utilizing the physical resources available in modern powerful servers, which reduces the total cost of ownership (TCO) for server deployments.

A hypervisor is the basic abstraction layer of software. The hypervisor performs low-level tasks such as CPU scheduling and is responsible for memory isolation for resident VMs. The hypervisor abstracts the hardware for the VMs. The hypervisor has no knowledge of networking, external storage devices, video, and so on.

Key components

This section gives you a high-level understanding of how XenServer works. See the following illustration for the key components of XenServer:



Architecture overview

Hardware

The hardware layer contains the physical server components, such as CPU, memory, network, and disk drives.

You need an Intel VT or AMD-V 64-bit x86-based system with one or more CPUs to run all supported guest operating systems. For more information about XenServer host system requirements, see System requirements.
For a complete list of XenServer certified hardware and systems, see the Hardware Compatibility List (HCL).

Xen Hypervisor

The Xen Project hypervisor is an open-source type-1 or bare-metal hypervisor. It allows many instances of an operating system or different operating systems to run in parallel on a single machine (or host). Xen hypervisor is used as the basis for many different commercial and open-source applications, such as: server virtualization, Infrastructure as a Service (IaaS), desktop virtualization, security applications, embedded, and hardware appliances.

XenServer is based on the Xen Project hypervisor and on top of that we provide extra features and support. XenServer uses version 4.13.4 of the Xen hypervisor.

Control domain

The **Control Domain**, also called Domain 0, or dom0, is a secure, privileged Linux VM that runs the XenServer management toolstack known as XAPI. This Linux VM is based on a CentOS 7.5 distribution. Besides providing XenServer management functions, dom0 also runs the physical device drivers for networking, storage, and so on. The control domain can talk to the hypervisor to instruct it to start or stop guest VMs.

Toolstack The **Toolstack**, or XAPI is the software stack that controls VM lifecycle operations, host and VM networking, VM storage, and user authentication. It also allows the management of XenServer resource pools.
XAPI provides the publicly documented management API, which is used by all tools that manage VMs, and resource pools. For more information, see the XenServer Management API.

Guest domain (VMs)

Guest domains are user-created virtual machines that request resources from dom0. For a detailed list of the supported distributions, see Supported Guests, Virtual Memory, and Disk Size Limits.

Full virtualization Full virtualization, or hardware-assisted virtualization uses virtualization extensions from the host CPU to virtualize guests. Fully virtualized guests do not require any kernel support. The guest is called a hardware virtual machine (HVM). HVM requires Intel VT or AMD-V hardware extensions for memory and privileged operations. XenServer uses Quick Emulator (QEMU) to emulate PC hardware, including BIOS, IDE disk controller, VGA graphic adaptor, USB controller, network adapter, and so on. To improve the performance of hardware-sensitive operations like disk or network access, HVM guests are installed with the XenServer tools. For more information, see PV on HVM.

HVM is commonly used when virtualizing an operating system such as Microsoft Windows where it is impossible to modify the kernel to make it virtualization aware.

PV on HVM PV on HVM is a mixture of paravirtualization and full hardware virtualization. The primary goal is to boost performance of HVM guests by using specially optimized paravirtualized drivers. This mode allows you to take advantage of the x86 virtual container technologies in newer processors for improved performance. Network and storage access from these guests still operate in PV mode, using drivers built in to the kernels.

Windows and Linux distributions are available in PV on HVM mode in XenServer. For a list of supported distributions using PV on HVM, see Guest Operating System Support.

XenServer VM Tools XenServer VM Tools (formerly Citrix VM Tools or XenServer PV Tools) provide high performance I/O services without the overhead of traditional device emulation.

- XenServer VM Tools for Windows consist of I/O drivers (also known as paravirtualized drivers or PV drivers) and the Management Agent.

  The I/O drivers contain front-end storage and network drivers, and low-level management interfaces. These drivers replace the emulated devices and provide high-speed transport between VMs and XenServer product family software.

  The Management Agent, also known as the guest agent, is responsible for high-level virtual machine management features. It provides full functionality to XenCenter (for Windows VMs).

  XenServer VM Tools for Windows must be installed on each Windows VM for the VM to have a fully supported configuration. A VM functions without the XenServer VM Tools for Windows, but performance will be significantly hampered when the I/O drivers (PV drivers) are not installed.

- XenServer VM Tools for Linux contain a guest agent that provides extra information about the VM to the host. Install the guest agent on each Linux VM to enable Dynamic Memory Control (DMC).

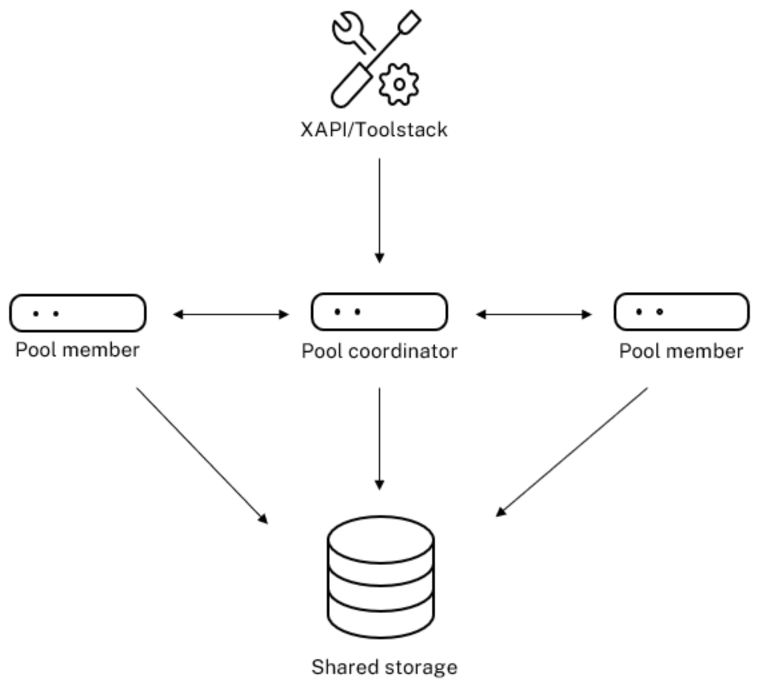
> **Note:**
>
> You cannot use the Dynamic Memory Control (DMC) feature on Red Hat Enterprise Linux 8, Red Hat Enterprise Linux 9, Rocky Linux 8, Rocky Linux 9, or CentOS Stream 9 VMs as these operating systems do not support memory ballooning with the Xen hypervisor.

For more information, see XenServer VM Tools.

Key concepts

Resource pool

XenServer allows you to manage multiple hosts and their connected shared storage as a single entity by using resource pools. Resource pools enable you to move and run virtual machines on different XenServer hosts. They also allow all hosts to share a common framework for network and storage. A pool can contain up to 64 hosts running the same version of XenServer software, at the same patch level, and with broadly compatible hardware. For more information, see Hosts and resource pools.

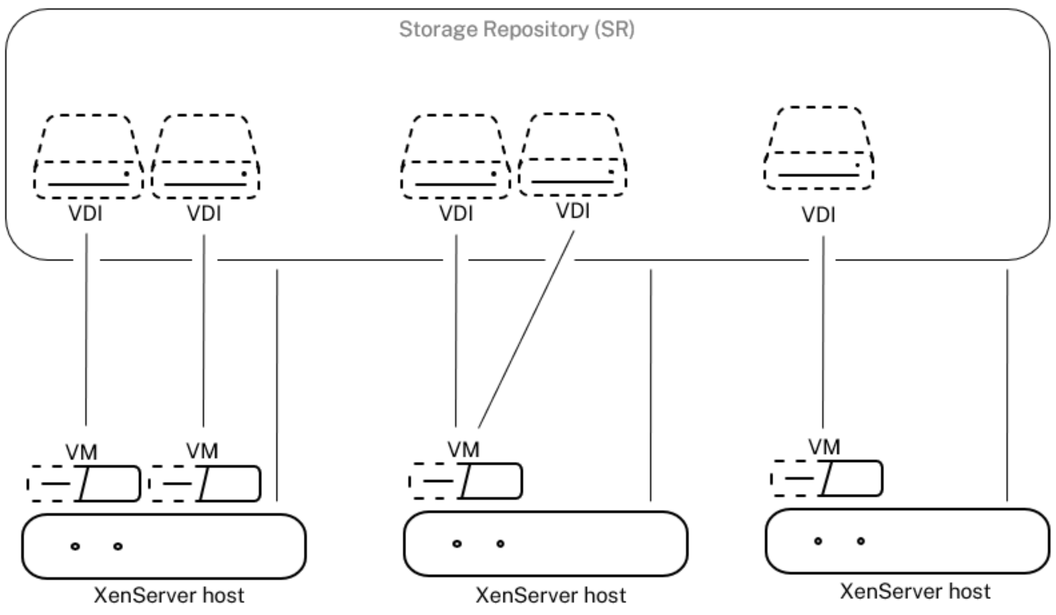


Resource pool overview

XenServer resource pool adopts a primary/secondaries architecture, implemented by XAPI. XAPI calls are forwarded from the pool coordinator (the primary) to pool members (the secondaries). Pool members make DB RPCs against the pool coordinator. The pool coordinator is responsible for coordination and locking resources within the pool, and processes all control operations. Pool members talk to the pool coordinator through HTTP and XMLRPC, but they can talk to each other (over the same channel) through mirror disks (storage migration)

Storage repository

XenServer storage targets are called storage repositories (SRs). A storage repository stores Virtual Disk Images (VDIs), which contains the contents of a virtual disk.
SRs are flexible, with built-in support for SATA, SCSI, NVMe, and SAS drives that are locally connected, and iSCSI, NFS, SAS, SMB, and Fibre Channel remotely connected. The SR and VDI abstractions allow advanced storage features such as thin provisioning, VDI snapshots, and fast cloning to be exposed on storage targets that support them.



Storage overview

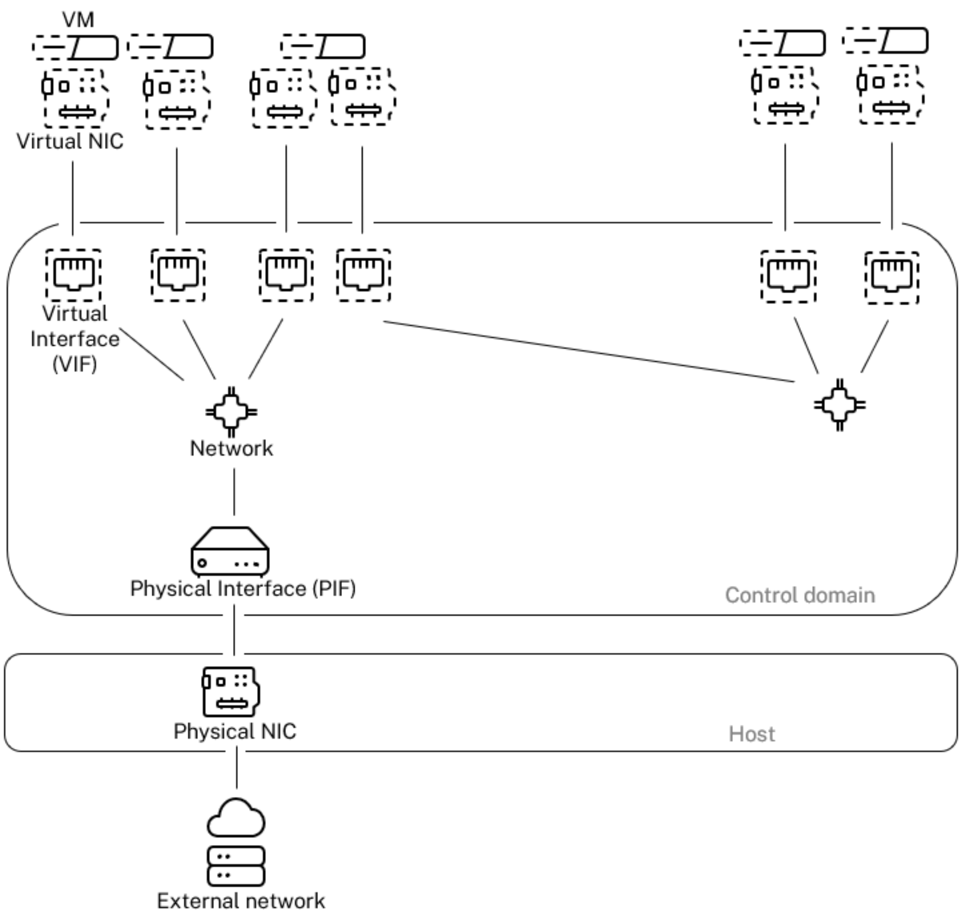
Each XenServer host can use multiple SRs and different SR types simultaneously. These SRs can be shared between hosts or dedicated to particular hosts. Shared storage is pooled between multiple hosts within a defined resource pool. A shared SR must be network-accessible to each host in the pool. All hosts in a single resource pool must have at least one shared SR. Shared storage cannot be shared between multiple pools.

For more information about how to operate with SRs, see Configure storage.

Networking

On an architecture level, there are three types of server-side software objects to represent networking entities. These objects are:

- A **PIF**, which is a software object used within dom0 and represents a physical NIC on a server. PIF objects have a name and description, a UUID, the parameters of the NIC that they represent, and the network and host they are connected to.
- A **VIF**, which is a software object used within in dom0 and represents a virtual NIC on a virtual machine. VIF objects have a name and description, a UUID, and the network and VM they are connected to.
- A **network**, which is a virtual Ethernet switch on a host used to route network traffic on a network host. Network objects have a name and description, a UUID, and the collection of VIFs and PIFs connected to them.



Networking overview

XenServer management APIs allow following operations:

- Configuration of networking options
- Control over the NIC to be used for management operations
- Creation of advanced networking features such as VLANs and NIC bonds

For more information about how to manage networks on XenServer, see Networking.

Related add-ons and applications

While Xen Hypervisor works at the core level, there are XenServer specific add-ons related hypervisor-agnostic applications and services available to make the virtualization experience complete.

- **XenCenter**

  A windows GUI client for VM management, implemented based on the management API. XenCenter provides a rich user experience to manage multiple XenServer hosts, resource pools, and the entire virtual infrastructure associated with them.

- **Workload Balancing (WLB)**

  An appliance that balances your pool by relocating virtual machines onto the best possible hosts for their workload in a resource pool. For more information, see Workload balancing (/en-us/xenserver/8/wlb.html).

- **Citrix Licensing Server**

  A Linux based appliance that XenCenter contacts to request a license for the specified server.

- **XenServer Conversion Manager**

  A virtual appliance that enables users to convert existing VMware virtual machines into XenServer virtual machines, with comparable networking and storage connectivity. For more information, see Conversion manager.

- **Citrix Provisioning**

  Provisioning Services that support PXE boot from common images. Used widely with Citrix Virtual Desktops and Citrix Virtual Apps. For more information, see Provisioning.

- **Citrix Virtual Desktops**

  A Virtual Desktop Infrastructure (VDI) product specialized to Windows desktops. Citrix Virtual Desktops uses XAPI to manage XenServer in a multi-host pool configuration. For more information, see Citrix Virtual Apps and Desktops.

Technical FAQs

March 21, 2024

Hardware

What are the minimum system requirements for running XenServer?

For the minimum system requirements for this release, see System requirements.

Do I need a system with a 64-bit x86 processor to run XenServer?

Yes. Either an Intel VT or AMD-V 64-bit x86-based system with one or more CPUs is required to run all supported guest operating systems.

For more information about host system requirements, see System requirements.

Do I need a system with hardware virtualization support?

You need a 64-bit x86 processor-based system that supports either Intel VT or AMD-V hardware virtualization technology in the processor and system firmware.

What systems are certified to run XenServer?

For a complete list of XenServer certified systems, see the Hardware Compatibility List (HCL).

Does XenServer support AMD Rapid Virtualization Indexing and Intel Extended Page Tables?

Yes. XenServer supports AMD Rapid Virtualization Indexing and Intel Extended Page Tables. Rapid Virtualization Indexing provides an implementation of nested tables technology used to further enhance the performance of the Xen hypervisor. Extended Page Tables provide an implementation of hardware-assisted paging used to further enhance the performance of the Xen hypervisor.

Can XenServer run on a notebook or desktop-class systems?

XenServer runs on many notebook or desktop-class systems that conform to the minimum CPU requirements. However, XenServer only supports systems that have been certified and listed on the Hardware Compatibility List (HCL).

You can choose to run on unsupported systems for demonstration and testing purposes. However, some features, such as power management capabilities, do not work.

Can XenServer be installed on SD or USB cards?

No. XenServer does not support using SD cards or USB cards for your XenServer installation.

We only support hardware that has been certified and listed on the Hardware Compatibility List (HCL).

Product limits

> **Note:**
>
> For a complete list of XenServer supported limits, see Configuration Limits.

What is the maximum size of memory that XenServer can use on a host system?

XenServer host systems can use up to 6 TB of physical memory.

How many processors can XenServer use?

XenServer supports up to 448 logical processors per host. The maximum number of logical processors supported differs by CPU.

For more information, see the Hardware Compatibility List (HCL).

How many virtual machines can run on XenServer concurrently?

The maximum number of virtual machines (VMs) supported to run on a XenServer host is 1000. For systems running more than 500 VMs, we recommend allocating at least 16 GB of RAM to dom0. For more information, see Change the amount of memory allocated to the control domain.

For any particular system, the number of VMs that can run concurrently and with acceptable performance depends on the available resources and the VM workload. XenServer automatically scales the amount of memory allocated to the control domain (dom0) based on the physical memory available.

> **Note:**
>
> If there are more than 50 VMs per host and the host physical memory is less than 48 GB, it might be advisable to override this setting. For more information, see Memory usage.

How many physical network interfaces does XenServer support?

XenServer supports up to 16 physical NIC ports. These NICs can be bonded to create up to 8 logical network bonds. Each bond can include up to 4 NICs.

How many virtual processors (vCPUs) can XenServer allocate to a VM?

XenServer supports up to 32 vCPUs per VM. The number of vCPUs that can be supported varies by the guest operating system.

> **Note:**
>
> Consult your guest OS documentation to ensure that you do not exceed the supported limits.

How much memory can XenServer allocate to a VM?

XenServer supports up to 1.5 TB per guest. The amount of memory that can be supported varies by the guest operating system.

> **Note:**
>
> The maximum amount of physical memory addressable by your operating system varies. Setting the memory to a level greater than the operating system supported limit can lead to performance issues within your guest.

How many virtual disk images (VDIs) can XenServer allocate to a VM?

XenServer can allocate up to 255 VDIs including a virtual DVD-ROM device per VM.

> **Note:**
>
> The maximum number of VDIs supported depends on the guest operating system. Consult your guest OS documentation to ensure that you do not exceed the supported limits.

How many virtual network interfaces can XenServer allocate to a VM?

XenServer can allocate up to 7 virtual NICs per VM. The number of virtual NICs that can be supported varies by the guest operating system.

Resource sharing

How are processing resources split between VMs?

XenServer splits processing resources between vCPUs using a fair-share balancing algorithm. This algorithm ensures that all VMs get their share of the processing resources of the system.

How does XenServer choose which physical processors it allocates to the VM?

XenServer doesn't statically allocate physical processors to any specific VM. Instead, XenServer dynamically allocates, depending on load, any available logical processors to the VM. This dynamic allocation ensures that processor cycles are used efficiently because the VM can run wherever there is spare capacity.

How are disk I/O resources split between the VMs?

XenServer uses a fair-share resource split for disk I/O resources between VMs. You can also provide a VM higher or lower priority access to disk I/O resources.

How are network I/O resources split between the VMs?

XenServer uses a fair-share resource split for network I/O resources between the VMs. You can also control the rate of outgoing data by using the Open vSwitch. For more information, see Control the rate of outgoing data (QoS).

Guest operating systems

Can XenServer run 32-bit operating systems as guests?

Yes. For more information, see Supported guest operating systems.

Can XenServer run 64-bit operating systems as guests?

Yes. For more information, see Supported guest operating systems.

Which versions of Microsoft Windows can run as guests on XenServer?

For a list of supported Windows guest operating systems, see Supported guest operating systems.

Which versions of Linux can run as guests on XenServer?

For a list of supported Linux guest operating systems, see Supported guest operating systems.

Can I run different versions of the supported operating systems or other unlisted operating systems?

We only support operating systems (OS) under OS vendor support. Although unsupported operating systems might continue to function, we might ask you to upgrade to a supported OS service pack before we can investigate any issues.

Applicable drivers might not be available for OS versions that are unsupported. Without the drivers, these OS versions do not function with optimized performance.

It's often possible to install other distributions of Linux. However, XenServer can only support the operating systems listed in Supported guest operating systems. We might ask you to switch to a supported OS before issues we can investigate any issues.

Does XenServer support FreeBSD, NetBSD, or any other BSD variants as a guest operating system?

XenServer doesn't support any BSD-based guest operating systems for general-purpose virtualization deployments. However, FreeBSD VMs running on XenServer have been certified for use in specific NetScaler products.

What are the XenServer VM Tools?

The XenServer VM Tools are software packages for Windows and Linux guest operating systems. For Windows operating systems, the XenServer VM Tools for Windows include high-performance I/O drivers (PV drivers) and the Management Agent.

For Linux operating systems, the XenServer VM Tools for Linux include a Guest Agent that provides additional information about the VM to the XenServer host.

For more information, see XenServer VM Tools.

Docker

Can I run Docker containers on my Linux VMs?

Yes. Docker is supported on Linux VMs that are hosted on XenServer.

Can I run Docker containers on my Windows VMs?

No. You cannot run Docker containers on a Windows VM that is hosted on XenServer. This restriction is because XenServer does not support nested virtualization for Windows VMs.

Does XenServer provide additional features for working with Docker?

No.

In previous releases of XenServer and Citrix Hypervisor, a Container Management supplemental pack was available that enabled you to manage your Docker containers through XenCenter. This feature has been removed.

XenCenter

For more information, see XenCenter.

Do I have to run XenCenter on a Windows computer?

Yes. The XenCenter management console runs on a Windows operating system. For information about the system requirements, see System requirements

If you don't want to run Windows, you can manage your XenServer hosts and pools by using the xe CLI or by using `xsconsole`, a system configuration console.

Can I log on to XenCenter using my Active Directory user accounts?

Yes. You can set up XenCenter login requests to use Active Directory on all editions of XenServer.

For more information, see Manage users.

Can I restrict access of certain functions within XenCenter to certain users?

Yes. The Role Based Access Control feature combined with Active Directory authentication can restrict access for users in XenCenter.

For more information, see Manage users.

Can I use a single XenCenter console to connect to multiple XenServer hosts?

Yes. You can use a single XenCenter console to connect to multiple XenServer host systems.

Can I use XenCenter to connect to multiple hosts running different versions of XenServer?

Depending on the version of XenServer - yes. XenCenter is backwards-compatible with Citrix Hypervisor 8.0 and later versions. However, note that only Citrix Hypervisor 8.2 CU 1 is receiving full support.

Can I use XenCenter to connect to multiple resource pools?

Yes. You can connect to multiple resource pools from a single XenCenter console.

How can I gain access to the console of a Linux VM?

The **Console** tab in XenCenter provides access to the text-based and graphical consoles of VMs running Linux operating systems. Before you can connect with the graphical console of a Linux VM, install and configure a VNC server and an X display manager on the VM.

XenCenter also enables you to connect to Linux VMs over SSH by using the **Open SSH Console** option on the **Console** tab of the VM.

How can I gain access to the console of a Windows VM?

XenCenter provides access to the emulated graphics for a Windows VM. If XenCenter detects remote desktop capabilities on the VM, XenCenter provides a quick connect button to launch a built-in RDP client that connects to the VM. Or, you can connect directly to your guests by using external remote desktop software.

Command line interface (CLI)

For more information, see Command-line interface.

Does XenServer include a CLI?

Yes. All editions of XenServer include a full command line interface (CLI) –known as `xe`.

Can I access the xe CLI directly on the host?

Yes. You can access the CLI by connecting a screen and keyboard directly to the host, or through a terminal emulator connected to the serial port of the host.

Can I access the xe CLI from a remote system?

Yes. XenServer ships the xe CLI, which can be installed on Windows and 64-bit Linux machines to control XenServer remotely. You can also use XenCenter to access the console of the host from the Console tab.

Can I use the xe CLI using my Active Directory user accounts?

Yes. You can log in using Active Directory on all editions of XenServer.

Can I restrict access the use of certain CLI commands to certain users?

Yes. You can restrict user access on the xe CLI.

VMs

For more information, see Manage virtual machines.

Can VMs created with VMware or Hyper-V run on XenServer?

Yes. You can export and import VMs using the industry-standard OVF format.

You can also convert VMs in batches using the XenServer Conversion Manager. Third-party tools are also available.

For more information, see Conversion Manager.

What types of installation media can I use to install a guest operating system?

You can install a guest operating system by using:

- A CD in the CD-ROM drive of the host
- A virtual CD-ROM drive using technology such as DRAC
- Placing ISO images on to a shared network drive
- Network installation, if supported by the specific guest.

For more information, see Manage Virtual Machines.

Can I make a clone of an existing VM?

Yes. Any VM created on XenServer can be cloned or converted into a VM template. A VM template can then be used to create more VMs.

Can VMs be exported from one version of XenServer and moved to another?

Yes. VMs exported from older versions of XenServer can be imported to a newer version.

Can I convert a VM from the open-source version of Xen to XenServer?

No.

Does XenServer provide disk snapshot capabilities for VMs?

Yes. XenServer supports using snapshots in all editions. For more information, see VM Snapshots.

Storage

For more information, see Storage.

What types of local storage can be used with XenServer?

XenServer supports local storage such as SATA, SAS, and NVMe.

What type of SAN/NAS storage can be used with XenServer?

XenServer supports Fibre Channel, FCoE, Hardware-based iSCSI (HBA), iSCSI, NFS, and SMB storage repositories.

For more information, see Storage and the Hardware Compatibility List.

Does XenServer support software-based iSCSI?

Yes. XenServer includes a built-in software-based iSCSI initiator (open-iSCSI).

What version of NFS is required for remote storage use?

XenServer requires NFSv3 or NFSv4 over TCP for remote storage use. XenServer currently does not support NFS over User Datagram Protocol (UDP).

Can I use software-based NFS running on a general-purpose server for remote shared storage?

Yes. Although we recommend using a dedicated NAS device with NFSv3 or NFSv4 with high-speed non-volatile caching to achieve acceptable levels of I/O performance.

Can I boot a XenServer host system from an iSCSI, Fibre Channel or FCoE SAN?

Yes. XenServer supports Boot from SAN using Fibre Channel, FCoE, or iSCSI HBAs.

Can I boot a XenServer host from UEFI?

Yes. XenServer supports booting from UEFI. However, UEFI Secure Boot is not supported for XenServer hosts.

Booting from BIOS is currently supported, but is deprecated and will be removed in a future release.

For more information, see Network boot installations

Does XenServer support Multipath I/O (MPIO) for storage connections?

Yes. We recommend using multipath for resilient storage connections.

Does XenServer support a software-based RAID implementation?

No. XenServer doesn't support software RAID.

Does XenServer support HostRAID or FakeRAID solutions?

No. XenServer doesn't support proprietary RAID-like solutions, such as HostRAID or FakeRAID.

Does XenServer support thin cloning of existing VMs?

Yes. Thin cloning is available on local disks formatted as EXT3/EXT4, in addition to NFS and SMB storage repositories.

Does XenServer support Distributed Replicated Block Device (DRBD) storage?

No. XenServer doesn't support DRBD.

Does XenServer support ATA over Ethernet?

No. XenServer doesn't support ATA over Ethernet-based storage.

Networking

For more information, see Networking

Can I create private networks that isolate groups of VMs?

Yes. You can create a private network on a single host for resident VMs.

Does XenServer support multiple physical network connections?

Yes. You can connect to or associate multiple physical networks that attach to different network interfaces on the physical host system.

Can VMs connect to multiple networks?

Yes. VMs can connect to any network available to the host.

Does XenServer support IPv6?

VMs hosted on XenServer can use any combination of IPv4 and IPv6 configured addresses.

However, XenServer doesn't support the use of IPv6 in its Control Domain (dom0). You can't use IPv6 for the host management network or the storage network. IPv4 must be available for the XenServer host to use.

Does XenServer support VLANs on a physical network interface?

Yes. XenServer supports assigning VM networks to specified VLANs.

Do XenServer virtual networks pass all network traffic to all VMs?

By default, XenServer network interfaces are non-promiscuous and a VM can only see traffic for that VM and broadcast traffic.

This behavior can be configured depending on the network stack that you are using.

- If you are using the Linux bridge as the network stack, your virtual network interfaces can be configured for promiscuous mode. This mode enables you to see all traffic on a virtual switch. For more information about promiscuous mode configuration, see the following Knowledge Center articles:

  - CTX116493 - How to Enable Promiscuous Mode on a Physical Network Card
  - CTX121729 - How to Configure a Promiscuous Virtual Machine in XenServer

  When you enable promiscuous mode on a virtual network interface, for a VM to make use of this configuration, you must also enable promiscuous mode within your VM.

- If you are using the Open vSwitch (OVS) as your network stack, it acts as a Layer 2 switch. A VM only sees traffic for that VM. Also, the switch-port locking in XenServer enables increased levels of isolation and security. OVS cannot be configured in promiscuous mode.

Does XenServer support bonding or teaming of physical network interfaces?

Yes. XenServer supports physical network interface bonding for failover and link aggregation with optional LACP support. For more information, see Networking.

Memory

How much memory is consumed by running XenServer?

Three components contribute to the memory footprint of a XenServer host.

1. The Xen hypervisor
2. The control domain on the host (dom0)
3. The XenServer Crash Kernel

The amount of memory required to run dom0 is adjusted automatically. By default, XenServer allocates 1 GiB plus 5% of the total physical memory to the control domain, up to an initial maximum of 8 GiB.

> **Note:**
>
> The amount of memory allocated to the Control Domain can be increased beyond the default amount.

In XenCenter, the **Xen** field in the **Memory** tab reports the memory used by the Control Domain, by the Xen hypervisor itself, and by the XenServer Crash Kernel. The amount of memory used by the hypervisor is larger for hosts with more memory.

For more information, see Memory usage

Does XenServer optimize VM memory usage?

Yes. XenServer uses Dynamic Memory Control (DMC) to automatically adjust the memory of running VMs. These adjustments keep the amount of memory allocated to each VM between specified minimum and maximum memory values, guaranteeing performance and permitting greater VM density.

For more information, see VM memory.

Resource pools

For more information, see Hosts and resource pools.

What is a resource pool?

A resource pool is a set of XenServer hosts managed as a unit. Typically, a resource pool shares some amount of networked storage to allow VMs to be rapidly migrated from one host to another within the pool.

Does XenServer require a dedicated host to manage a resource pool?

No. A single host in the pool must be specified as the pool coordinator. The pool coordinator controls all administrative activities required on the pool. This design means that there is no external single point of failure. If the pool coordinator fails, other hosts in the pool continue to operate, and the resident VMs continue to run as normal. If the pool coordinator cannot come back online, XenServer promotes one of the other hosts in the pool to coordinator to regain control of the pool.

This process is automated with the High Availability feature. For more information, see High availability.

Where is the configuration data for a resource pool stored?

A copy of the configuration data is stored on every host in the resource pool. If the current pool coordinator fails, this data enables any host in the resource pool to become the new pool coordinator.

What types of configurations can be made at the resource pool level?

Shared remote storage and networking configurations can be made at the resource pool level. When a configuration is shared on the resource pool, the coordinator system automatically propagates configuration changes to all the member systems.

Are new host systems added to a resource pool automatically configured with shared settings?

Yes. Any new host systems added to a resource pool automatically receive the same configurations for shared storage and network settings.

Can I use different types of CPUs in the same XenServer resource pool?

Yes. We recommend that the same CPU type is used throughout the pool (homogeneous resource pool). However, it is possible for hosts with different CPU types to join a pool (heterogeneous), provided the CPUs are from the same vendor.

For more information, see Hosts and resource pools.

For updated information about the support for feature masking for specific CPU types, see Hardware Compatibility List.

Live Migration (formerly XenMotion)

For more information, see Migrate VMs.

Can I move a running VM from one host to another?

With live migration you can move running VMs when hosts share storage (in a pool).

Also, storage live migration allows migration between hosts that do not share storage. VMs can be migrated within or across pools.

High availability

For more information, see High availability.

Does XenServer offer high availability features?

Yes. If high availability is enabled, XenServer continually monitors the health of the hosts in a pool. If high availability detects that a host is impaired, the host is automatically shut down. This action allows for VMs to be restarted safely on an alternative healthy host.

Does XenServer high availability support local storage?

No. If you want to use high availability, shared storage is required. This shared storage enables VMs to be relocated if a host fails. However, high availability allows VMs that are stored on local storage to be marked for automatic restart when the host recovers after a reboot.

Can I use high availability to automatically sequence the restart of recovered VMs?

Yes. High availability configuration allows you to define the order that VMs are started. This capability enables VMs that depend on one another to be sequenced automatically.

Performance metrics

Do the XenServer management tools collect performance information?

Yes. XenServer provides detailed monitoring of performance metrics. These metrics include CPU, memory, disk, network, C-state/P-state information, and storage. Where appropriate, these metrics are available on a per-host and a per-VM basis. Performance metrics are available directly (exposed as Round Robin Databases), or can be accessed and viewed graphically in XenCenter or other third-party applications. For more information, see Monitor and manage your deployment.

How are XenServer performance metrics gathered?

Data for the XenServer performance metrics are collected from various sources. These sources include the Xen hypervisor, Dom0, standard Linux interfaces, and standard Windows interfaces such as WMI.

Does XenCenter display performance metrics in real time?

Yes. XenCenter displays real-time performance metrics on the **Performance** tab for each running VM and for the XenServer host. You can customize the metrics that are displayed.

Does XenCenter store and display historic performance metrics?

Yes. XenServer keeps performance metrics from the last year (with decreasing granularity). XenCenter provides a visualization of these metrics in real-time graphical displays.

Installation

For more information, see Install.

Does XenServer install on top of systems that are already running an existing operating system?

No. XenServer installs directly on bare-metal hardware, avoiding the complexity, overhead, and performance bottlenecks of an underlying operating system.

Can I upgrade an existing XenServer installation to a newer version?

Yes. If you are running a supported version of XenServer you can upgrade to a newer version of XenServer instead of doing a fresh installation. For more information, see Upgrade.

Can I upgrade from an out-of-support version of Citrix Hypervisor or XenServer to this version?

If your existing version of Citrix Hypervisor or XenServer is no longer in support, you cannot upgrade or update to the latest version of XenServer. Only upgrades from Citrix Hypervisor 8.2 Cumulative Update 1 are supported.

For more information, see Upgrade.

How much local storage does XenServer require for installation on the physical host system?

XenServer requires a minimum of 46 GB of local storage on the physical host system.

Can I use PXE to do a network installation of XenServer on the host system?

Yes. You can install XenServer on the host system by using PXE. You can also automatically install XenServer using PXE by creating a pre-configured answer file.

Does the Xen hypervisor run on Linux?

No. Xen is a Type 1 hypervisor that runs directly on the host hardware ("bare metal"). After the hypervisor loads, it starts the privileged management domain –the control domain (dom0), which contains a minimal Linux environment.

Where does XenServer get its device driver support?

XenServer uses the device drivers available from the Linux kernel. As a result, XenServer runs on a wide variety of hardware and storage devices. However, we recommend that you use certified device drivers.

For more information, see the Hardware Compatibility List.

Licensing

For information about XenServer licensing, see Licensing.

Technical Support

For more information about support, see Support.

Does XenServer provide direct technical support for XenServer?

Yes. For more information, visit the XenServer support pages.

Do I have to buy a XenServer technical support contract at the same time as I buy XenServer?

No. A technical support contract is included in your license purchase. For information about the level of support we provide for Premium and Standard Edition customers, visit the XenServer support pages.

What level of support does my license entitle me to?

When you buy a XenServer per-socket license, you also get the benefit of our Technical Support services. For more information about levels of support, see https://xenserver.com/support.

Can I get support if my hosts are running in Trial Edition?

If you are a Trial Edition user, you are not eligible for support. However, we value your feedback: Provide feedback.

Are there alternative channels for getting support for XenServer?

Yes. There are several alternative channels for getting technical support for XenServer. You can also use Citrix Knowledge Center or contract with authorized XenServer partners who offer technical support services.

Does XenServer provide technical support for the open-source Xen project?

No. XenServer doesn't provide technical support for the open-source Xen project. For more information, visit http://www.xen.org/.

Can I open a technical support incident with XenServer if I'm experiencing a non-technical issue?

No. Raise any non-technical issues through Citrix Customer Service. For example, issues to do with software maintenance, licensing, administrative support, and order confirmation.

Licensing overview

March 25, 2024

> **Important:**
>
> If you are using XenServer to run your Citrix Virtual Apps and Desktops workloads, you must have a Premium Edition license. For more information about getting a XenServer license, see https://xenserver.com/buy.
> This requirement is a change of behavior since the previous version of Citrix Hypervisor/XenServer. For more information, see the licensing FAQ.

> Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

XenServer 8 is available in the following editions:

- **Premium Edition** is our premium offering, optimized for desktop, server, and cloud workloads. In addition to the features available in the Standard Edition, the Premium Edition offers the following features:
  - Dynamic Workload Balancing
  - GPU virtualization with NVIDIA vGPU and Intel GVT-g
  - Thin provisioning for shared block storage devices
  - Support for SMB storage
  - Direct Inspect APIs
  - Export pool resource data
  - In-memory read caching
  - PVS-Accelerator
  - Enablement for Citrix Virtual Desktops tablet mode
  - Changed block tracking
  - IGMP snooping
  - USB pass-through
  - SR-IOV network support
  - Monitor host and dom0 resources with NRPE
  - Monitor host and dom0 resources with SNMP
- **Standard Edition** is our entry-level commercial offering. It has a range of features for customers who want a robust and high-performing virtualization platform, but don't require the premium features of Premium Edition. Meanwhile, they still want to benefit from the assurance of comprehensive support and maintenance.
- **Trial Edition** currently showcases our Premium Edition feature set on a restricted size pool (maximum three hosts), but doesn't allow rolling pool upgrade through XenCenter.

For more information, see https://www.xenserver.com/editions.

	Premium Edition	Standard Edition	Trial Edition
Premium feature set	Yes		Yes
Rolling pool upgrade through XenCenter	Yes	Yes	
Continuous functional and security updates	Yes	Yes	Yes

	Premium Edition	Standard Edition	Trial Edition
Maximum pool size	64 (16 with GFS2 SRs)	64 (16 with GFS2 SRs)	3

XenServer is licensed on a *per-socket* basis. Allocation of licenses is managed centrally and enforced by a standalone Citrix License Server (physical or virtual) in the environment. For more information about XenServer licensing, see the Licensing FAQ.

Licensing your hosts and pools

XenServer uses the same licensing process as some Citrix products. To use XenServer Premium Edition or XenServer Standard Edition, you require a valid license to be installed on a Citrix License Server and assigned to your XenServer host. This process is covered in detail in the Licensing guide for XenServer.

> **Note:**
>
> XenServer does not currently support licensing hosted on Citrix Cloud. An on-premises Citrix License Server is required.

To license your hosts, you need the following items:

- A license
- A Citrix License Server
- A XenServer host
- XenCenter

1. Install Citrix License Server

You can download the Citrix License Server for Windows from the Citrix Licensing downloads page.

Install the Citrix License Server on a Windows system according to the instructions in the Citrix Licensing documentation.

The Windows system that you install your Citrix License Server on can be a Windows VM hosted in your XenServer pool.

XenServer operates with a 'grace' license until the License Server can boot. This behavior means, after you have licensed the XenServer hosts in your pool, if you reboot the host that has the Citrix License Server running on it, a grace period is applied until the License Server is restarted.

2. Download license files

Download a license file that is tied to the case-sensitive host name of your Citrix License Server and contains a large enough number of per-socket licenses to share across the hosts you want to license.

For information about purchasing a XenServer license, see https://xenserver.com/buy.

For more information about getting your license files, see the Citrix Licensing product documentation.

3. Add the license file to the Citrix License Server

Use the Citrix Licensing Manager to install the licenses on your Citrix License Server. For more information, see the Citrix Licensing product documentation.

4. Apply the licenses to hosts in your resource pool

You can allocate licenses hosted on a Citrix License Server to your XenServer hosts and pools by using XenCenter or the xe CLI.

Apply a license to all the hosts using XenCenter A: Follow this procedure to apply a license:

1. On the **Tools** menu, click **License Manager**.
2. Select the pool or hosts that you want to license, and then click **Assign License**.
3. In the **Apply License** dialog, specify the **Edition** type to assign to the host.
4. Type the host name or IP address of the Citrix License Server.
5. Click **OK**.

Apply a license to hosts or pools by using the xe CLI To apply a license to a single host, run the `host-apply-edition` command:

```
    xe host-apply-edition edition=premium-per-socket|standard-per-
        socket  \

        license-server-address=<license_server_address> host-uuid=<
            uuid_of_host>  \

        license-server-port=<license_server_port>
<!--NeedCopy-->
```

To apply a license to all hosts in a pool, run the `pool-apply-edition` command:

```
    xe pool-apply-edition edition=premium-per-socket|standard-per-
        socket  \

        license-server-address=<license_server_address> pool-uuid=<
            uuid_of_pool>  \

        license-server-port=<license_server_port>
<!--NeedCopy-->
```

Licensing FAQ

April 8, 2024

This article contains frequently asked questions about licensing your XenServer hosts and pools.

- General questions
- Citrix Virtual Apps and Desktops
- Citrix License Servers
- Licensing a XenServer pool
- More information

General questions

Q: Where can I buy a XenServer license?

A: You can learn about buying a XenServer License at http://www.xenserver.com/buy.

Q: How do I apply a XenServer license?

A: XenServer requires a License Server. After licensing XenServer, you are provided with a .LIC license access code. Install this license access code on your Citrix License Server.

When you assign a license to a XenServer host, XenServer contacts the specified Citrix License Server and requests a license for the specified hosts. If successful, a license is checked out and the License Manager displays information about the license the hosts are licensed under.

Q: How many licenses do I need to license my resource pool?

A: XenServer is licensed on a *per-CPU socket* basis. For a pool to be considered licensed, all XenServer hosts in the pool must be licensed. XenServer only counts populated CPU sockets.

You can use the Citrix License Server to view the number of available licenses displayed in the *License Administration Console Dashboard*.

Q: Do I need a per-socket license for sockets that are not populated?

A: No, only populated CPU sockets are counted toward the number of sockets to be licensed.

Q: Do I lose my virtual machine (VM) when my license expires?

A: No, you do not lose any VMs or their data.

Q: What happens if I have a licensed pool and the License Server becomes unavailable?

A. If your license has not expired and the License Server is unavailable, you receive a *grace period* of 30 days at the licensing level that was applied previously.

After the grace period, if your license is still unavailable, your pool changes to a Trial Edition pool and only those features included in Trial Edition are available. Your pool remains the same size, but if it has three or more hosts, you cannot add any more hosts to it.

Q: How do I get a license to evaluate XenServer?

A: With Trial Edition, you can install XenServer without a license and trial all the Premium Edition features in a reduced-size pool. For more information, see XenServer editions.

Q: Can I use XenServer 8 without a license?

A: Yes. XenServer 8 Trial Edition makes XenServer available without a license. The edition provides all of the Premium Edition features in a reduced-size pool of up to three hosts. For more information, see XenServer editions.

Q: Can I use Citrix Cloud to apply a license to XenServer?

No, XenServer does not currently support licensing hosted on Citrix Cloud. To license XenServer, you require a License Server. For more information, see Licensing your hosts and pools.

Q: What level of support does my license entitle me to?

When you buy a XenServer per-socket license, you also get the benefit of our Technical Support services. For more information about levels of support, see https://xenserver.com/support.

If you are a Trial Edition user, you are not eligible for support. However, we value your feedback: Provide feedback.

For more information, see Technical Support questions and Support

Q: I am upgrading to XenServer 8 from a previous Citrix Hypervisor version with a per-socket license. Do I have to do anything?

A: No. You can upgrade your hosts to XenServer 8 using the previously bought per-socket licenses, provided Customer Success Services is valid at least until Mar 1, 2024.

If you have renewed your Customer Success Services after the original purchase, you might need to refresh the license file on the License Server to ensure it displays the Customer Success Services eligibility.

Q: I am upgrading to XenServer 8 from a previous Citrix Hypervisor version with a Citrix Virtual Apps and Desktops license. Do I have to do anything?

A: Yes. Before attempting to upgrade to XenServer 8, you must obtain XenServer Premium Edition licenses, import them into Citrix License Server, and assign them to the XenServer hosts that are currently using Citrix Virtual Apps and Desktops licenses.

For more information, see Citrix Virtual Apps and Desktops questions.

Q: I am moving from the preview version of XenServer 8 to the GA version of XenServer 8. Do I have to do anything?

A: Maybe. This depends on your license.

- If you are using a Premium or Standard Edition license or are using XenServer 8 preview without a license (Trial Edition), your XenServer hosts require no licensing changes.
- If you are using a Citrix Virtual Apps and Desktops license, your XenServer hosts show this license as deprecated. You must get a XenServer Premium Edition license to continue to run your Citrix Virtual Apps and Desktops workloads on XenServer.

For more information, see Citrix Virtual Apps and Desktops questions.

Citrix Virtual Apps and Desktops

Q: What edition of XenServer do I need to run Citrix Virtual Apps and Desktops workloads on XenServer?

A: To run your Citrix Virtual Apps and Desktops workloads on XenServer, you must have a Premium Edition per-socket license for XenServer. Premium Edition delivers many advanced features that make XenServer a highly optimized hypervisor platform for your workload. For more information, see XenServer editions.

For more information about getting a XenServer license, see https://xenserver.com/buy.

Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

Q: I am a Citrix Virtual Apps and Desktops or Citrix DaaS customer moving from an earlier version of Citrix Hypervisor to XenServer 8. Do I have to do anything?

A: Yes. Before attempting to upgrade to XenServer 8, you must obtain XenServer Premium Edition licenses, import them into Citrix License Server, and assign them to the XenServer hosts that are currently using Citrix Virtual Apps and Desktops licenses.

For more information about getting a XenServer license, see https://xenserver.com/buy. Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

This behavior is different to that of earlier versions of Citrix Hypervisor and XenServer. Citrix Hypervisor 8.2 Cumulative Update 1 and earlier were available as an entitlement to Citrix Virtual Apps and Desktops customers.

Q: I am a Citrix Service Provider licensed for Citrix Virtual Apps and Desktops or Citrix DaaS. Can I use this license for XenServer when I upgrade to XenServer 8?

A: No. Before attempting to upgrade to XenServer 8, you must obtain XenServer Premium Edition licenses, import them into Citrix License Server, and assign them to the XenServer hosts that are currently using Citrix Virtual Apps and Desktops licenses.

For more information about getting a XenServer license, see https://xenserver.com/buy. Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

This behavior is different to that of earlier versions of Citrix Hypervisor and XenServer. Citrix Hypervisor 8.2 Cumulative Update 1 and earlier were available as an entitlement to Citrix Virtual Apps and Desktops customers.

Q: I am a customer with a Citrix DaaS subscription. Am I entitled to use XenServer 8?

A: No. You must obtain XenServer Premium Edition licenses, import them into Citrix License Server, and assign them to your XenServer hosts.

For more information about getting a XenServer license, see https://xenserver.com/buy. Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

This behavior is different to that of earlier versions of Citrix Hypervisor and XenServer. Citrix Hypervisor 8.2 Cumulative Update 1 and earlier were available as an entitlement to Citrix DaaS customers.

Q: What are the constraints on the use of the XenServer advanced virtualization management capabilities delivered as part of Citrix Virtual Apps and Desktops?

A: To run Citrix Virtual Apps and Desktops workloads on XenServer, you need a Premium Edition license. Premium Edition delivers many advanced features that make XenServer a highly optimized hypervisor platform for your workload.

For information about the advanced capabilities that XenServer provides for Citrix Virtual Apps and Desktops, see Using XenServer with Citrix products.

For more information about getting a XenServer license, see https://xenserver.com/buy. Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

Citrix License Servers

Q: Which License Servers can I use with XenServer?

A: You can use the Citrix License Server software version 11.16 or later on a server running Microsoft Windows.

In previous releases, we supported a Linux-based License Server virtual appliance. This product is no longer supported. If you are using the License Server virtual appliance with an existing pool, migrate to the latest version of Citrix License Server for Windows before upgrading to XenServer 8.

Q: How do I import my license onto the Citrix License Server?

A: For information on importing a license file, see the Citrix Licensing documentation.

Q: Can I run the License Server on my XenServer pool?

A: Yes. You can install the Citrix License Server software on a Windows VM.

XenServer operates with a 'grace' license until the License Server is able to boot. This behavior means, after you have licensed the XenServer hosts in your pool, and you reboot the host that has the Citrix License Server running on it, a grace period is applied to that host until the License Server is restarted.

Q: Can I use the Windows version of the Citrix License Server with XenServer?

A: Yes.

Q: Can I install Licenses for Citrix products on the Citrix License Server software installed on Windows?

A: Yes, you can license Citrix products by using the Citrix License Server software installed on Windows. For more information, see Licensing on the Citrix Product Documentation website.

Licensing a XenServer pool

Q: How do I apply a license to all the hosts using XenCenter?

A: Follow this procedure to apply a license:

1. On the **Tools** menu, click **License Manager**.
2. Select the Pool or Hosts you would like to license, and then click **Assign License**.
3. In the **Apply License** dialog, specify the **Edition** type to assign to the host, and type the host name or IP address of the License Server.

Q: Can I apply a license without using XenCenter?

A: Yes, you can use the xe CLI. Run the `host-apply-edition` command. For example, enter the following to license a host:

```
    xe host-apply-edition edition=enterprise-per-socket|standard-per-
       socket  \

        license-server-address=<license_server_address> host-uuid=<
           uuid_of_host>  \

        license-server-port=<license_server_port>
<!--NeedCopy-->
```

To license a pool, use the `pool-apply-edition` command. For example, enter the following to license a pool:

```
    xe pool-apply-edition edition=enterprise-per-socket|standard-per-
       socket  \

        license-server-address=<license_server_address> host-uuid=<
           uuid_of_host>  \

        license-server-port=<license_server_port>
<!--NeedCopy-->
```

Q: How can I discover the license status of my hosts and pools?

A: XenCenter displays the license type of a host or pool.

To see the license type of a host or pool, select that host or pool in the tree view. XenCenter displays the license status in the title bar for that host or pool, after the host or pool name.

You can also go to the **General** tab of the host and find the license type in the **License Details** section.

To find the license type of a host by using the command line, run the following command in the console of a host in your pool:

```
xe host-license-view host\_uuid=<UUID> | grep sku\_marketing\_name
```

More information

- For more information about the XenServer 8 release, see XenServer 8 Release Notes.
- To access XenServer 8 product documentation, see XenServer 8 Product Documentation.
- For an overview of the XenServer product, see Technical Overview.
- For support information, see https://xenserver.com/support.

Install

April 8, 2024

XenServer installs directly on bare-metal hardware avoiding the complexity, overhead, and performance bottlenecks of an underlying operating system.

XenServer uses the device drivers available from the Linux kernel. As a result, XenServer can run on a wide variety of hardware and storage devices. However, ensure that you use certified device drivers. For more information, see the Hardware Compatibility List (HCL).

> **Important:**
>
> The XenServer host must be installed on a dedicated 64-bit x86 server. Do not install any other operating system in a dual-boot configuration with the XenServer host. This configuration is not supported.

This section is primarily aimed at system administrators who want to set up XenServer hosts on physical servers. It contains procedures to guide you through the installation or upgrade process. It also contains information about troubleshooting problems that might occur during installation and points you to extra resources.

Before you start

Depending on your environment, the installation method to use to get your hosts and pools to the latest version of XenServer 8 differs.

- If you already have Citrix Hypervisor 8.2 Cumulative Update 1 installed on your hosts and pools, Upgrade from Citrix Hypervisor 8.2 Cumulative Update 1.
- If you already installed XenServer 8 (including during its preview period), you cannot upgrade or update from the installation ISOs. Instead, apply the latest level of frequent updates through XenCenter. For more information, see Update XenServer 8.
- If you are already running any other version of XenServer or Citrix Hypervisor on your hosts and pools, upgrading from these versions is not supported. Do a fresh installation of XenServer 8.
- If you are installing XenServer for the first time on your hosts and pools, do a fresh installation of XenServer 8.

Installation methods

XenServer 8 can be installed in one of the following ways:

Fresh installation If you are creating a fresh installation of XenServer 8:

- Use the **XenServer 8 Installation ISO** file. You can download this file from the XenServer download page.

- Review the information in System Requirements, Licensing XenServer, and Installing XenServer and XenCenter before installing XenServer.

Upgrade If you are upgrading from Citrix Hypervisor 8.2 Cumulative Update 1 to XenServer 8:

- Use the **XenServer 8 Installation ISO** file. You can download this file from the XenServer download page.
- Review the information in System Requirements, Licensing XenServer, and Upgrading from an existing version before upgrading XenServer.

The installer presents the option to upgrade when it detects a previously installed version of XenServer. The upgrade process follows the first-time installation process, but several setup steps are bypassed. The existing settings are retained, including networking configuration, system time and so on.

You cannot upgrade directly from out-of-support versions of XenServer or Citrix Hypervisor to XenServer 8. Instead, perform a fresh installation.

Update Existing installations of XenServer 8 receive the latest updates through the frequent update mechanism. For more information, see Update XenServer 8.

Supported boot modes

XenServer supports booting hosts using either UEFI or BIOS boot mode. UEFI Secure Boot is not currently available for XenServer hosts.

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. You can still install your XenServer 8 hosts in BIOS boot mode. However, doing so can prevent you from upgrading your XenServer 8 hosts to a future version of XenServer. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

The server boot mode changes how you initiate the installation process. After the installer starts, the installation process is the same for both boot modes.

Install the XenServer host

This procedure takes you through doing a manual installation from local media. For information about other types of installation - such as network installation, unattended installation, or boot from SAN, see Other installation scenarios.

> **Tip:**
>
> Throughout the installation, quickly advance to the next screen by pressing **F12**. Use the **Tab** key

> to move between elements and **Space** or **Enter** to select. Press **F1** for general help.

To install a XenServer host:

1. Back up any data you want to preserve. Installing XenServer overwrites data on any hard drives that you select to use for the installation.
2. Boot the computer from the installation media:
   - To install XenServer host from a bootable USB:
     a) Create a bootable USB from the XenServer installation ISO. Ensure that the tool does not alter the contents of the ISO file.
        - On Linux, you can use the `dd` command to write the ISO to a USB. For example, `dd if=<path_to_source_iso> of=<path_to_destination_usb>`.
        - On Windows, you can use Rufus. Ensure that you select **Write in DD Image mode**. If this is not selected, Rufus can alter the contents of the ISO file and cause it not to boot.

     b) Insert the bootable USB drive into the target system.

     c) Restart the system.

     d) Go into the boot menu.

     e) Change the settings to boot the system from the USB.

     (If necessary, see your hardware vendor documentation for information on changing the boot order)
   - To install XenServer host from a CD/DVD:
     a) Burn the XenServer installation ISO file to a CD/DVD.

     b) Insert the bootable CD/DVD into the CD/DVD drive on the target system.

     c) Restart the system.

     d) Go into the boot menu.

     e) Change the settings to boot the system from the CD/DVD.

     (If necessary, see your hardware vendor documentation for information on changing the boot order)
   - To install XenServer host from virtual media:
     a) Go to the virtual console of your system.

     b) Insert the XenServer installation ISO file as virtual media.

     c) Restart the system.

   d) Go into the boot menu.

   e) Change the settings to boot the system from the virtual media.

      (If necessary, see your hardware vendor documentation for information on changing the boot order)

   - For information about network installation, see Other installation scenarios.

3. Following the initial boot messages and the **Welcome to XenServer** screen, select your key map (keyboard layout) for the installation.

   > **Note:**
   >
   > If a System Hardware warning screen is displayed and hardware virtualization assist support is available on your system, see your hardware manufacturer for BIOS upgrades.

4. At the **Welcome to XenServer Setup** screen, XenServer offers the following options:

   - **To load a device driver press <F9>**

     XenServer ships with a broad driver set that supports most modern server hardware configurations. However, you might need to apply driver disks (a type of supplemental pack) in order to be able to perform the XenServer installation. If you have been provided with any additional essential device drivers, press F9. The installer steps you through loading the necessary drivers.

     > **Warning:**
     >
     > You cannot install other types of supplemental packs at this point in the installation process. You can install them along with additional driver disks near the end of the installation process.

   - **To set up advanced storage classes press <F10>**

     If you have done the necessary configuration in your network infrastructure, you can configure the XenServer installation to boot from software FCoE (deprecated). Press F10 and follow the instructions displayed on the screen to set up software FCoE. For more information, see Other installation scenarios.

   After you have completed any steps you require on this page, select **OK** to proceed.

5. Scroll through and read the XenServer End User Agreement (EUA). Select **Accept EUA** to proceed.

   If you choose not to accept the EUA, you cannot continue with the installation.

6. Select the appropriate action from the list. This list always includes:

- **Perform clean installation**: Choose this option to continue with a fresh installation.

Depending on the state of your server, you might also see the following options:

- **Upgrade**: If the installer detects a previously installed version of XenServer or Citrix Hypervisor, it offers the option to upgrade. For information about upgrading your XenServer host, see Upgrading from an existing version.
- **Restore**: If the installer detects a previously created backup installation, it offers the option to restore XenServer from a backup.

Make your selection, and choose **OK** to proceed.

7. If you have multiple local hard disks, choose a primary disk for the installation. Select **OK**.
8. Choose which disks that you want to use for virtual machine storage. View information about a specific disk by pressing **F5**. Select **OK**.
9. If you selected disks that all have 512 byte blocks, you have the option to use thin provisioning to optimize the use of available storage. Select **Enable thin provisioning** to allow the local SR of the host to be used for local caching of VM VDIs. Citrix Virtual Desktops and DaaS users are recommended to select this option for local caching to work properly. For more information, see Storage.

   > **Note:**
   >
   > If you selected disks that are 4KB native, the virtual machine storage is automatically configured for a large disk block size.

   Select **OK** to proceed.
10. Select your installation media source.

    - To install from a USB, CD, or virtual media, select **Local media**.
    - To install from the network, select **HTTP** or **FTP** or **NFS**.

    Select **OK** to proceed.
11. If you select HTTP or FTP or NFS in the previous step, set up networking so that the installer can connect to the XenServer installation media files:

    a) If the computer has multiple NICs, select one of them to be used to access the XenServer installation media files. Choose **OK** to proceed.

    b) Choose **Automatic configuration (DHCP)** to configure the NIC using DHCP, or Static configuration to configure the NIC manually. If you choose **Static configuration**, enter details as appropriate.

    c) Provide VLAN ID if you have your installation media present in a VLAN network.

   d) If you choose **HTTP** or **FTP**, provide the URL for your HTTP or FTP repository, and a user name and password, if appropriate.

      If you choose **NFS**, provide the server and path of your NFS share.

   e) Choose **OK** to proceed.

   For more information about setting up your installation media on NFS, FTP, or HTTP, see Other installation scenarios.

12. Indicate if you want to verify the integrity of the installation media. If you select **Verify installation source**, the SHA256 checksum of the packages is calculated and checked against the known value. Verification can take some time. Make your selection and choose **OK** to proceed.
13. Set and confirm a root password, which XenCenter uses to connect to the XenServer host. You also use this password (with user name "root") to log into **xsconsole**, the system configuration console.

    > **Note:**
    >
    > XenServer root passwords must contain only ASCII characters.

14. Set up the primary management interface that XenCenter uses to connect to and manage the host.

    If your computer has multiple NICs, select the NIC that you want to use for management. Choose **OK** to proceed.

15. Configure the management interface with the following options:

    - Choose **Automatic configuration (DHCP)** to configure the NIC using DHCP.
    - Choose **Static configuration** to configure the NIC manually. Provide an IP address, subnet mask, and gateway.
    - Select **Use VLAN** to have the management interface on a VLAN network. Provide the VLAN ID.

    > **Note:**
    >
    > To be part of a pool, XenServer hosts must have static IP addresses or be DNS addressable. When using DHCP, ensure that a static DHCP reservation policy is in place.

16. Specify the host name and the DNS configuration.

    a) In the **Hostname Configuration** section, select from the following options:

       - Choose **Automatically set via DHCP** to have the DHCP server provide the host name along with the IP address.
       - Choose **Manually specify** to define the host name yourself. Enter the host name for the server in the field provided.

> **Note:**
>
> If you manually specify the host name, enter a short host name and *not the fully qualified domain name (FQDN)*. Entering an FQDN can cause external authentication to fail, or the XenServer host might be added to Active Directory with a different name.

a) In the **DNS Configuration** section, select from the following options:

- Choose **Automatically set via DHCP** to get name service configuration using DHCP.
- Select **Manually specify** to define the DNS servers yourself. Enter the IP addresses of your primary (required), secondary (optional), and tertiary (optional) DNS servers in the fields provided.

Select **OK** to proceed.

17. Select your time zone by geographical area and city. You can type the first letter of the desired locale to jump to the first entry that begins with this letter. Choose **OK** to proceed.
18. Specify how you want the XenServer host to determine local time. XenServer offers the following options:

    - **Using NTP**: Select this option to use the NTP protocol to set your server time. On the next screen, configure it in one of the following ways:

      – Select **NTP is configured by my DHCP server** to have your network provide the NTP server host name or IP address.

      – Manually enter at least one NTP server name or IP address.

      Choose **OK** to proceed.

    - **Manual time entry**: Select this option to set the date and time manually.

      On the next screen, enter the current time in UTC.

      Choose **OK** to proceed.

    > **Note:**
    >
    > XenServer assumes that the time setting on the server is the current time in UTC.

    Choose **OK** to proceed.

19. Select **Install XenServer**.

    The installation process starts. This process might take some minutes.

20. The next screen asks if you want to install any supplemental packs (including driver disks).

> **Note:**
>
> If you have already loaded a driver disk during initial installation, you might be prompted to reinsert the driver disk so that the driver can be installed onto disk. At this point, reinsert the driver disk to ensure that your XenServer instance contains the new driver.

- If you want to install any supplemental packs or driver disks provided by your hardware supplier, choose **Yes**.

  a) You are prompted to insert the supplemental pack. Eject the XenServer installation media, and insert the supplemental pack media.

  b) Choose **OK**.

  c) Select **Use media** to proceed with the installation.

  d) Repeat for each pack to be installed.

- If you do not want to install a supplemental pack, choose **No**.

  The installation process completes. This process can take a few minutes.

21. When prompted by the **Installation Complete** screen, eject the installation media (if installing from USB or CD).
22. Select **OK** to reboot the host.

After the host reboots, XenServer displays **xsconsole**, a system configuration console. To access a local shell from **xsconsole**, press **Alt+F3**; to return to **xsconsole**, press **Alt+F1**.

> **Note:**
>
> Make note of the IP address displayed. Use this IP address when you connect XenCenter to the XenServer host.

Install XenCenter

XenCenter must be installed on a Windows machine that can connect to the XenServer host through your network. Ensure that .NET framework version 4.8 or above is installed on this system.

To install XenCenter:

1. Download the installer for the latest version of XenCenter from the XenServer download page.
2. Launch the installer `.msi` file.
3. Follow the **Setup** wizard, which allows you to modify the default destination folder and then to install XenCenter.

For more information about using XenCenter, see the XenCenter documentation.

Connect XenCenter to the XenServer host

To connect XenCenter to the XenServer host:

1. Launch XenCenter. The program opens to the **Home** tab.
2. Click the **Add New Server** icon.
3. Enter the IP address of the XenServer host in the **Server** field. Type the root user name and password that you set during XenServer installation. Click **Add**.
4. The first time you add a host, the **Save and Restore Connection State** dialog box appears. This dialog enables you to set your preferences for storing your host connection information and automatically restoring host connections.

   If you later want to change your preferences, you can do so from the XenCenter main menu, select **Tools** and then **Options**. The **Options** dialog box opens. Select the **Save and Restore** tab and set your preferences. Click **OK** to save your changes.

License your XenServer hosts

Your newly installed XenServer hosts can run in Trial Edition without a license. This edition restricts your pool size and doesn't allow rolling pool upgrade through XenCenter. For more information, see http://www.xenserver.com/editions.

For information about licensing your XenServer hosts, see Licensing XenServer.

Other installation scenarios

February 13, 2024

In addition to a standard manual installation process, XenServer provides the ability to perform various other types of installations, including the following:

- Network installations using PXE boot
- Unattended installations
- Setting up the host to boot from SAN
- Configuring host multipathing

Supported boot modes

XenServer supports booting hosts using either UEFI or BIOS boot mode. UEFI Secure Boot is not currently available for XenServer hosts.

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. You can still install your XenServer 8 hosts in BIOS boot mode. However, doing so can prevent you from upgrading your XenServer 8 hosts to a future version of XenServer. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

The server boot mode changes how you initiate the installation process. After the installer starts, the installation process is the same for both boot modes.

When upgrading your XenServer hosts, ensure that the upgrade uses the same boot mode as the initial install.

Network installation

If the server that you want to install on has a PXE boot-enabled Ethernet card, you can use this feature to do a network installation with PXE boot.

Using PXE boot to install from the network involves the following steps:

- Copy the installer files to a TFTP server and configure your TFTP and DHCP servers for PXE boot installation. The method for doing this depends on your boot mode: BIOS or UEFI.
- Host your installation media on NFS, FTP, or HTTP. Only the installer files are accessed from the TFTP server. The XenServer files to be installed on the server are hosted on an NFS, FTP, or HTTP server. Alternatively, after you start the install through PXE boot, you can complete it from local media hosted on the target server.
- Create an answer file for unattended installation. You can instead choose to do an attended installation and step through the installer manually.
- Start the installation process.

> **Note:**
>
> PXE boot is not supported over a tagged VLAN network. Ensure that the VLAN network you use for PXE boot is untagged.

Configure your TFTP and DHCP servers

Before you set up the XenServer installation media, configure your TFTP and DHCP servers. The following sections contain information on how to configure your TFTP server for PXE boot with BIOS or UEFI. Consult your vendor documentation for general setup procedures.

Configure your TFTP server for PXE boot with BIOS

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

Host the installer files on a TFTP server and configure your TFTP server to enable PXE booting with BIOS boot mode. This configuration is used to start the installation process.

1. In your TFTP root directory (for example, `/tftpboot`), create a directory called `xenserver`.
2. From the XenServer installation media, copy the `mboot.c32` and `pxelinux.0` files from the `/boot/pxelinux` directory of your installation media to the TFTP root directory.

   > **Note:**
   >
   > We strongly recommend using `mboot.c32` and `pxelinux.0` files from the same source (for example, from the same XenServer installation ISO).

3. From the XenServer installation media, copy the files to the new `xenserver` directory on the TFTP server:

   - `install.img` from the root directory
   - `vmlinuz` from the `/boot` directory
   - `xen.gz` from the `/boot` directory

4. In the TFTP root directory (for example, `/tftpboot`), create a directory called `pxelinux.cfg`.
5. In the `pxelinux.cfg` directory, create your configuration file called **`default`**.

   The content of this file depends on how you want to configure your PXE boot environment and on the values that are appropriate for your servers.

   - **Example: Unattended install** This example configuration performs an unattended installation using the answer file at the URL specified:

```
    default xenserver-auto
    label xenserver-auto
        kernel mboot.c32
        append xenserver/xen.gz dom0_max_vcpus=1-16 \
            dom0_mem=max:8192M com1=115200,8n1 \
            console=com1,vga ---  xenserver/vmlinuz \
            console=hvc0 console=tty0 \
            answerfile=<http://pxehost.example.com/
                answer_file> \
            answerfile_device=<device> \
            install ---  xenserver/install.img
<!--NeedCopy-->
```

> **Note:**
>
> To specify which network adapter to use to retrieve the answer file, include the `answerfile_device=ethX` or `answerfile_device=MAC` parameter and specify either the Ethernet device number or the MAC address of the device.

For more information about using an answer file, see Create an answer file for unattended installation.

- **Example: Manual install** This example configuration starts an installation that boots from the TFTP server and requires manual responses:

```
    default xenserver
    label xenserver
        kernel mboot.c32
        append xenserver/xen.gz dom0_max_vcpus=1-16 \
        dom0_mem=max:8192M com1=115200,8n1 \
        console=com1,vga ---  xenserver/vmlinuz \
        console=hvc0 console=tty0 \
        ---  xenserver/install.img
<!--NeedCopy-->
```

For more information about PXE configuration file contents, see the SYSLINUX website.

Next step: Host your installation media on NFS, FTP, or HTTP. In addition to the TFTP and DHCP servers, you require an NFS, FTP, or HTTP server to house the XenServer files that are installed on your server.

Configure your TFTP server for PXE boot with UEFI Host the installer files on a TFTP server and configure your DHCP and TFTP servers to enable PXE booting with UEFI boot mode. This configuration is used to start the installation process.

1. In the TFTP root directory (for example, `/tftpboot`), create a directory called `EFI/xenserver`.
2. Copy the following files from the XenServer installation media to the new `EFI/xenserver` directory on the TFTP server:

    - `grubx64.efi` from the `/EFI/xenserver` directory
    - `install.img` from the root directory
    - `vmlinuz` from the `/boot` directory
    - `xen.gz` from the `/boot` directory

3. Configure your DHCP server to provide `/EFI/xenserver/grubx64.efi` as the boot file.
4. In the `EFI/xenserver` directory on the TFTP server, create the `grub.cfg` file.

The content of this file depends on how you want to configure your PXE boot environment and on the values that are appropriate for your servers.

- **Example: Unattended install** This example configuration performs an unattended installation using the answer file at the URL specified:

```
menuentry "XenServer Install (serial)" {

    multiboot2 /EFI/xenserver/xen.gz dom0_max_vcpus=1-16
        dom0_mem=max:8192M com1=115200,8n1 console=com1,vga
    module2 /EFI/xenserver/vmlinuz console=hvc0 console=tty0
        answerfile_device=eth0 answerfile=http://<ip_address
        >/<path_to_answer_file> install
    module2 /EFI/xenserver/install.img
 }

<!--NeedCopy-->
```

> **Note:**
>
> To specify which network adapter to use to retrieve the answer file, include the `answerfile_device=ethX` or `answerfile_device=MAC` parameter and specify either the Ethernet device number or the MAC address of the device.

  For more information about using an answer file, see Create an answer file for unattended installation.

- **Example: Manual install** This example configuration starts an installation that boots from the TFTP server and requires manual responses:

```
menuentry "XenServer Install (serial)" {

    multiboot2 /EFI/xenserver/xen.gz dom0_max_vcpus=1-16
        dom0_mem=max:8192M com1=115200,8n1 console=com1,vga
    module2 /EFI/xenserver/vmlinuz console=hvc0 console=tty0
    module2 /EFI/xenserver/install.img
 }

<!--NeedCopy-->
```

Next step: Host your installation media on NFS, FTP, or HTTP. In addition to the TFTP and DHCP servers, you require an NFS, FTP, or HTTP server to house the XenServer files that are installed on your server.

Host your installation media on NFS, FTP, or HTTP

The TFTP server hosts the files needed to start the installer, but the files to be installed are hosted on an NFS, FTP, or HTTP server.

You can also use files hosted on NFS, FTP, or HTTP to complete an installation that has been started from local media on your server.

1. On the HTTP, FTP, or NFS server, create a directory from which the XenServer installation media can be exported through HTTP, FTP, or NFS.
2. Copy the entire contents of the XenServer installation media to the newly created directory on the HTTP, FTP, or NFS server. This directory is your installation repository.

   > **Note:**
   >
   > When copying the XenServer installation media, ensure that you copy the file `.treeinfo` to the newly created directory.
   >
   > If you are using IIS to host the installation media, ensure that double escaping is enabled on IIS before extracting the installation ISO on it.

Next step:

- If you are completing an unattended installation: Create an answer file for unattended installation.
- If you are using PXE boot to start a manual installation: Start the network installation.

Create an answer file for unattended installation

To perform installations in an unattended fashion, create an XML answer file.

Contain all nodes within a root node named *installation*. When constructing your answer file, refer to the Answer file reference.

Here is an example answer file:

```xml
<?xml version="1.0"?>
    <installation srtype="ext">
        <primary-disk>sda</primary-disk>
        <guest-disk>sdb</guest-disk>
        <guest-disk>sdc</guest-disk>
        <keymap>us</keymap>
        <root-password>mypassword</root-password>
        <source type="url">http://pxehost.example.com/xenserver/</
           source>
        <script stage="filesystem-populated" type="url">
          http://pxehost.example.com/myscripts/post-install-script
        </script>
        <admin-interface name="eth0" proto="dhcp" />
        <timezone>Europe/London</timezone>
    </installation>
<!--NeedCopy-->
```

Next step: Start the network installation.

Automated upgrades with an answer file You can also perform automated upgrades by changing the answer file appropriately.

1. Set the `mode` attribute of the `installation` element to `upgrade`.
2. Specify the disk on which the existing installation lives with the `existing-installation` element.
3. Leave the `primary-disk` and `guest-disk` elements unspecified.

For example:

```
<?xml version="1.0"?>
<installation mode="upgrade">
    <existing-installation>sda</existing-installation>
    <source type="url">http://pxehost.example.com/xenserver/</source>
    <script stage="filesystem-populated" type="url">
        http://pxehost.example.com/myscripts/post-install-script
    </script>
</installation>
<!--NeedCopy-->
```

Answer file reference The following is a summary of the elements. All node values are text, unless otherwise stated. Required elements are indicated.

`<installation>` **Required?** Yes

Description: The root element that contains all the other elements.

Attributes:

`srtype`

The attribute `srtype` can have one of the following values: `lvm`, `ext`, or `xfs`:

- `lvm` - set the local storage type to LVM.
- `ext` - set the local storage type to EXT4. This enables local caching for Citrix Virtual Desktops to work properly. For more information, see Storage.
- `xfs` - set the local storage type to XFS. This option also allows you to create local storage devices with 4 KB physical blocks without requiring a logical block size of 512 bytes.

To enable thin provisioning, you can specify the `srtype` attribute as `ext` or `xfs`. If you do not specify the `srtype` attribute, the default value for `srtype` is `lvm`. If you do not specify the `srtype` attribute but you configure a 4 KB native disk for local storage in your answer file, the default value is `xfs`.

> **Note:**
>
> You cannot use Local LVM or Local EXT3/EXT4 storage types with 4 KB physical blocks. If you attempt to specify `lvm` or `ext` for the `srtype` attribute whilst configuring 4 KB physical blocks,

> your answer file configuration is rejected as incompatible.

`mode`

To change the installation type to upgrade, specify a `mode` attribute with the value `upgrade`. If this attribute is not specified, the installer performs a fresh installation and overwrites any existing data on the server.

`<driver-source>` **Required?** Yes

Description: The source of a supplemental pack containing device drivers to be loaded by the installer and included after installation of the main repository.

Attributes: None

`<primary-disk>` **Required?** Yes

> **Note:**
>
> Deprecated for upgrade scenarios.

Description: The name of the storage device where the control domain is installed. This element is equivalent to the choice made on the *Select Primary Disk* step of the manual installation process.

Attributes: You can specify a `guest-storage` attribute with possible values `yes` and `no`.
For example: `<primary-disk guest-storage="no">sda</primary-disk>`

The default value is `yes`. If you specify `no`, you can automate an installation scenario where no storage repository is created. In this case, specify no guest-disk keys.

`<guest-disk>` **Required?** No

Description: The name of a storage device to be used for storing guests. Use one of these elements for each extra disk.

Attributes: None

`<ntp>` **Required?** Yes

Description: Specifies the source for NTP servers. If the `<ntp>` element is not specified, the default shall be `manual` if `<ntp-server>` is specified, `dhcp` if using DHCP, otherwise **`default`**.

Attributes:

The attribute `source` can have one of the following values: `dhcp`, **`default`**, `manual`, or `none`.

- `dhcp` - use NTP servers from DHCP
- **`default`** - use default NTP servers
- `manual` - use provided NTP servers, in this case at least one `<ntp-server>` entry must be specified
- `none` - NTP is disabled

If `source` is `dhcp`, **`default`**, or `none`, do not specify `<ntp-server>`.

`<ntp-server>` **Required?** No

Description: Specifies one or more NTP servers. To be used only with the `ntp` element and the attribute `manual`.

Attributes: None

`<keymap>` **Required?** No

Description: The name of the key map to use during installation. `<keymap>us</keymap>` The default value, `us` is considered if you do not specify a value for this element.

Attributes: None

`<root-password>` **Required:** No

Description: The desired root password for the XenServer host. If a password is not provided, a prompt is displayed when the host is first booted.

Attributes: You can specify a `type` that is either `hash` or `plaintext`

For example:

```
<root-password type="hash">hashedpassword</root-password>
<!--NeedCopy-->
```

The hashed value can use any hash type supported by `crypt(3)` in `glibc`. The default hash type is SHA-512.

You can use the following Python code to generate a hashed password string to include in the answer file:

```
python -c 'import crypt; print(crypt.crypt("mypasswordhere", crypt.
    mksalt(crypt.METHOD_SHA512)))'
<!--NeedCopy-->
```

`<source>` **Required:** Yes

Description: The location of the uploaded XenServer installation media or a Supplemental Pack. This element can occur multiple times.

Attributes: The attribute `type` can have one of the following values: `url`, `nfs`, or `local`.

If the value is `local`, leave the element empty. For example,

```
<source type="url">http://server/packages</source>
<source type="local" />
<source type="nfs">server:/packages</source>
<!--NeedCopy-->
```

`<script>` **Required:** No

Description: Where the post-install-script is located.

Attributes:

The attribute `stage` can have one of the following values: `filesystem-populated`, `installation-start`, or `installation-complete`.

- When the value `filesystem-populated` is used, the script runs just before the root file system is unmounted (for example, after installation/upgrade, initrds already built, and so on). The script receives an argument that is the mount point of the root file system.
- When the value `installation-start` is used, the script runs before starting the main installation sequence, but after the installer has initialized, loaded any drivers, and processed the answerfile. The script does not receive any arguments.
- When the value `installation-complete` is used, the script runs after the installer has finished all operations (and hence the root file system is unmounted). The script receives an argument that has a value of zero if the installation completed successfully, and is non-zero if the installation failed for any reason.

The attribute `type` can have one of the following values: `url`, `nfs`, or `local`.

If the value is `url` or `nfs`, put the URL or NFS path in the PCDATA. If the value is `local`, leave the PCDATA empty. For example,

```
<script stage="filesystem-populated" type="url">
    http://prehost.example.com/post-install-script
</script>
<script stage="installation-start" type="local">
    file:///scripts/run.sh
</script>
<script stage="installation-complete" type="nfs">
    server:/scripts/installation-pass-fail-script
</script>
```

```
<!--NeedCopy-->
```

> **Note:**
>
> If a local file is used, ensure that the path is absolute. This generally means that the `file://` prefix is followed by another forward slash, and then the complete path to the script.

`<admin-interface>` **Required:** Sometimes

> **Note:**
>
> Required during install/reinstall but not during upgrade or restore.

Description: The single network interface to be used as the host administration interface.

Attributes:

Specify one of the following attributes:

- `name` - The name of your network interface, for example `eth0`.
- `hwaddr` - The MAC address of your network interface, for example `00:00:11:aa:bb:cc`.

The attribute `proto` can have one of the following values: `dhcp` or `static`.

If you specify `proto="static"`, you must also specify all of these child elements:

Child elements

- `<ipaddr>`: The IP address
- `<subnet>`: The subnet mask
- `<gateway>`: The gateway

`<timezone>` **Required:** No

Description: The timezone in the format used by the TZ variable, for example Europe/London, or America/Los_Angeles. The default value is `Etc/UTC`.

`<name-server>` **Required:** No

Description: The IP address of a nameserver. Use one of these elements for each nameserver you want to use.

`<hostname>` **Required:** No

Description: Specify this element if you want to manually set a host name.

`<ntp-server>` **Required:** No

Description: Specify one or more NTP servers.

Start the network installation

After setting up the network servers required for a PXE boot installation, complete the following steps on the server that you are installing onto:

1. Start the system and enter the boot menu (**F12** in most BIOS programs).
2. Select to boot from your Ethernet card.
3. The system then PXE boots from the installation source you set up and the installation script starts.
    - If you have set up an answer file, the installation proceeds unattended.
    - If you have decided to do a manual installation, provide information when prompted. For more information, see Install.

Boot from SAN

Boot-from-SAN environments offer several advantages, including high performance, redundancy, and space consolidation. In these environments, the boot disk is on a remote SAN and not on the local host.

The following types of boot-from-SAN configuration are supported:

- HBA and hardware Fibre Channel
- Software FCoE (deprecated)
- Software boot from iSCSI

For a fully redundant boot-from-SAN environment, you must configure multiple paths for I/O access. For more information, see Enable multipathing.

HBA and hardware Fibre Channel

This type of boot-from-SAN deployment depends on SAN-based disk arrays with either hardware Fibre Channel or HBA iSCSI adapter support on the host. The host communicates with the SAN through a host bus adapter (HBA). The HBA's BIOS contains the instructions that enable the host to find the boot disk.

All of the configuration to set up boot from SAN through hardware Fibre Channel or an HBA adapter is done in your network infrastructure before you install XenServer on your servers. For information about how to complete this set up, see the documentation provided by the vendor.

After your network infrastructure is correctly set up, enable multipathing on your servers during the XenServer installation process. For more information, see Enable multipathing. Proceed with the installation as normal.

Software FCoE (deprecated)

You can boot a XenServer host from an FCoE SAN by using a software FCoE stack.

For this type of boot-from-SAN deployment, before installing your XenServer host, manually complete the configuration required to expose a LUN to the host. This manual configuration includes configuring the storage fabric and allocating LUNs to the public worldwide name (PWWN) of your SAN. After you complete this configuration, the available LUN is mounted to the CNA of the host as a SCSI device. The SCSI device can then be used to access the LUN as if it were a locally attached SCSI device. When you configure the FCoE fabric, do not use VLAN 0. The XenServer host cannot find traffic that is on VLAN 0.

For information about configuring the physical switch and the array to support FCoE, see the documentation provided by the vendor.

After your network infrastructure is correctly set up, enable multipathing on your servers during the XenServer installation process. For more information, see Enable multipathing. Proceed with the installation as normal.

During a manual installation of XenServer, you are given the option on the **Welcome to XenServer Setup** screen to set up advanced storage classes. Press **F10** and follow the instructions displayed on the screen to set up software FCoE.

Software boot from iSCSI

The software-boot-from-iSCSI feature enables customers to install and boot XenServer from SAN using iSCSI. Using this feature, XenServer can be installed to, booted from, and run from a LUN provided by an iSCSI target. The iSCSI target is specified in the iSCSI Boot Firmware Table. This capability allows the root disk to be attached through iSCSI. This boot disk can be located on the same target that provides an SR.

To use this feature, ensure that your environment meets the following requirements:

- The network interface or interfaces dedicated to iSCSI boot must be separate from the management interfaces and interfaces used for VM traffic.
- Storage (iSCSI targets) must be on a separate Layer 3 (IP) network to all other network interfaces with IP addresses on the host.
- Do not use tagged VLAN for the network interfaces dedicated to the iSCSI boot targets.

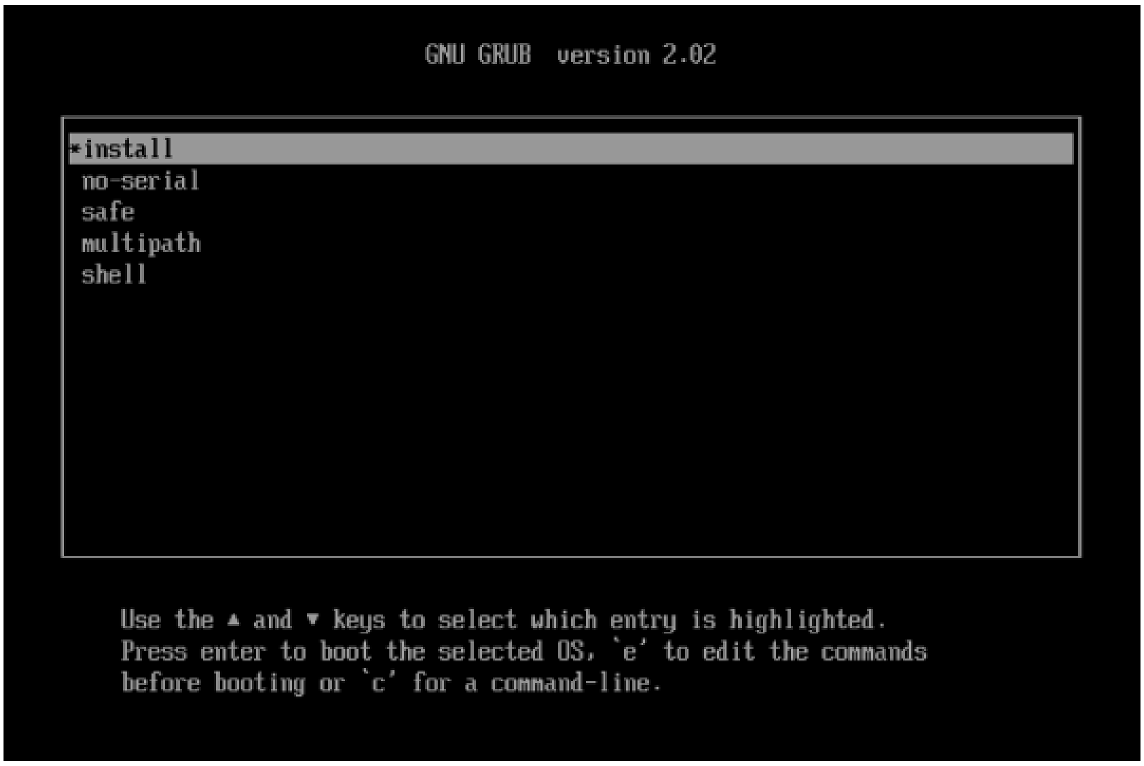
- We recommend that you enable multipathing on your servers.

To configure the software-boot-from-iSCSI feature, you must add the `use_ibft` parameter to your boot parameters. How you add this parameter depends on your boot mode and the type of installation you are doing.

Enable the software-boot-from-iSCSI feature on a UEFI boot server during an installation from local media

1. Boot the computer from the installation media. For more information, see Install the XenServer host.

   Following the initial boot messages, you see a GRUB menu. This menu is shown for 5 seconds.

   ```
   GNU GRUB  version 2.02

   *install
    no-serial
    safe
    multipath
    shell

   Use the ▲ and ▼ keys to select which entry is highlighted.
   Press enter to boot the selected OS, `e' to edit the commands
   before booting or `c' for a command-line.
   ```

2. Use the cursor keys to select an installation option:

   - For a single path LUN, select **install**.
   - For a multipathed LUN, select **multipath** (recommended).

3. Press the `e` key to edit the commands before booting.

4. Edit the line starting with the following:

   ```
   module2 /EFI/xenserver/vmlinuz ...
   <!--NeedCopy-->
   ```

Using the cursor keys, edit this line to include `use_ibft` at the end:

```
module2 /EFI/xenserver/vmlinuz ... use_ibft
<!--NeedCopy-->
```

5. Press **Enter**.
6. Continue your XenServer host installation process as normal.

Enable the software-boot-from-iSCSI feature on a BIOS boot server during an installation from local media

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

1. Boot the computer from the installation media. For more information, see Install the XenServer host.

   Following the initial boot messages, you see the **Welcome to XenServer** screen.

2. At the boot prompt, enter `menu.c32`.
3. Use the cursor keys to select an installation option:
   - For a single path LUN, select **install**.
   - For a multipathed LUN, select **multipath**.
4. Press the tab key.
5. Edit the line ending with the following:

   ```
   ---  /install.img
   <!--NeedCopy-->
   ```

   Using the cursor keys, edit this line to read:

   ```
   use_ibft ---  /install.img
   <!--NeedCopy-->
   ```

6. Press **Enter**.
7. Continue your XenServer host installation process as normal.

Enable the software-boot-from-iSCSI feature on a UEFI boot server during a PXE boot installation When installing using PXE, ensure that you add the keyword **use_ibft** in the kernel parameters. If multipathing is required, you must add **device_mapper_multipath=enabled**.

For example:

```
menuentry "XenServer Install (serial)" {

    multiboot2 /EFI/xenserver/xen.gz dom0_max_vcpus=1-16 dom0_mem=max
       :8192M com1=115200,8n1 console=com1,vga
    module2 /EFI/xenserver/vmlinuz console=hvc0 console=tty0
       answerfile_device=eth0 answerfile=http://<ip_address>/<
       path_to_answer_file> install use_ibft device_mapper_multipath=
       enabled
    module2 /EFI/xenserver/install.img
     }

<!--NeedCopy-->
```

For more information about setting up PXE boot, see Configure your TFTP server for PXE boot with UEFI.

Enable the software-boot-from-iSCSI feature on a BIOS boot server during a PXE boot installation

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

When installing using PXE, ensure that you add the keyword **use_ibft** in the kernel parameters. If using multipathing (recommended) you must add **device_mapper_multipath=enabled**.

For example:

```
default xenserver-auto
label xenserver-auto
    kernel mboot.c32
    append xenserver/xen.gz dom0_max_vcpus=1-16 \
    dom0_mem=max:8192M com1=115200,8n1 \
    console=com1,vga ---  xenserver/vmlinuz \
    console=hvc0 console=tty0 \
    answerfile=<http://pxehost.example.com/answer_file> \
    answerfile_device=<device> \
    use_ibft device_mapper_multipath=enabled ---  xenserver/install.img
<!--NeedCopy-->
```

For more information about setting up PXE boot, see Configure your TFTP server for PXE boot with BIOS.

Enable multipathing

For a fully redundant boot-from-SAN environment, you must configure multiple paths for I/O access. To do so, ensure that the root device has multipath support enabled.

For information about whether multipath is available for your SAN environment, consult your storage vendor or administrator.

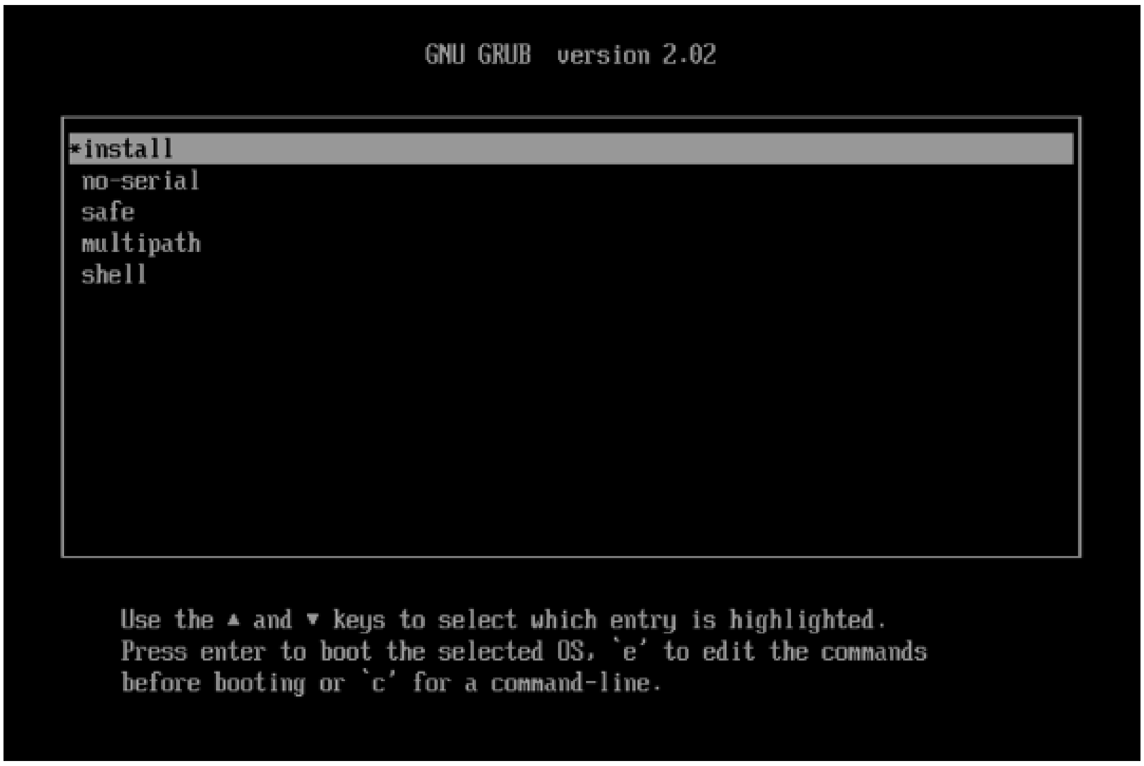
> **Warning:**
>
> Multipath settings are *not* inherited during the upgrade process. When upgrading using the ISO or network boot, follow the same instructions as used in the following installation process to ensure that `multipath` is correctly configured.

If you have multiple paths available, enable multipathing in your XenServer deployment while initializing the installation process. How you enable multipathing depends on your boot mode and the type of installation you are doing.

Enable multipathing on a UEFI boot server during a manual installation

1. Boot the computer from the installation media. For more information, see Install the XenServer host.

   Following the initial boot messages, you see a GRUB menu. This menu is shown for 5 seconds.



```
                     GNU GRUB  version 2.02

 *install
  no-serial
  safe
  multipath
  shell

      Use the ▲ and ▼ keys to select which entry is highlighted.
      Press enter to boot the selected OS, `e' to edit the commands
      before booting or `c' for a command-line.
```

2. On the GRUB menu, choose `multipath` and press **Enter**.

The XenServer installation process configures the XenServer host, which boots from a remote SAN with multipathing enabled.

Enable multipathing on a BIOS boot server during a manual installation

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

1. Boot the computer from the installation media. For more information, see Install the XenServer host.

   Following the initial boot messages, you see the **Welcome to XenServer** screen.

2. At the welcome screen, press **F2** to select **Advanced** install.
3. At the boot prompt, enter `multipath`.

The XenServer installation process configures the XenServer host, which boots from a remote SAN with multipathing enabled.

Enable multipathing on a UEFI boot server during an unattended installation To enable file system multipathing during PXE installation, add `device_mapper_multipath=enabled` to your configuration file.

For example:

```
menuentry "XenServer Install (serial)" {

    multiboot2 /EFI/xenserver/xen.gz dom0_max_vcpus=1-16 dom0_mem=max
       :8192M com1=115200,8n1 console=com1,vga
    module2 /EFI/xenserver/vmlinuz console=hvc0 console=tty0
       answerfile_device=eth0 answerfile=http://<ip_address>/<
       path_to_answer_file> install use_ibft device_mapper_multipath=
       enabled
    module2 /EFI/xenserver/install.img
     }

<!--NeedCopy-->
```

For more information about setting up PXE boot, see Configure your TFTP server for PXE boot with UEFI.

Enable multipathing on a BIOS boot server during an unattended installation

> **Note:**
>
> Booting XenServer hosts in BIOS mode is now deprecated. We recommend that you install your XenServer 8 hosts by using UEFI boot mode.

To enable file system multipathing during PXE installation, add `device_mapper_multipath=enabled` to your configuration file.

For example:

```
default xenserver-auto
label xenserver-auto
    kernel mboot.c32
    append xenserver/xen.gz dom0_max_vcpus=1-16 \
    dom0_mem=max:8192M com1=115200,8n1 \
    console=com1,vga ---  xenserver/vmlinuz \
    console=hvc0 console=tty0 \
    answerfile=<http://pxehost.example.com/answer_file> \
    answerfile_device=<device> \
    device_mapper_multipath=enabled \
    install ---  xenserver/install.img
<!--NeedCopy-->
```

For more information about setting up PXE boot, see Configure your TFTP server for PXE boot with BIOS.

Install supplemental packs

Supplemental packs are used to modify and extend the capabilities of XenServer by installing software into the control domain (dom0). For example, an OEM partner might want to ship XenServer with a set of management tools that require SNMP agents to be installed. You can install a supplemental pack either during initial XenServer installation, or any time afterwards on a running XenServer instance.

When installing supplemental packs during XenServer installation, unpack each supplemental pack into a separate directory on a web server.

You can install the supplemental pack in one of the following ways:

- During an interactive installation, when you are prompted to install supplemental packs, specify the URL to the supplemental pack media.
- If you are using an answer file for your install, add an additional `<source>` element to specify the location of the supplemental pack.

Install driver disks

You can install a driver disk using one of the following methods:

- By using XenCenter (recommended)
- During a clean XenServer installation
- By using the xe CLI

For information on how to install a driver disk by using XenCenter, see Install driver disks. For information on how to install a driver disk during a clean XenServer installation, see Install the XenServer host.

After installing the driver, restart your server for the new version of the driver to take effect. As with any software update, we advise you to back up your data before installing a driver disk.

Install a driver disk by using the xe CLI

Perform the following steps to install the driver disk remotely using the xe CLI:

1. Download the driver disk to a known location on a computer that has the remote xe CLI installed.
2. Extract the contents of the zip file.

   For the next step, ensure that you use the driver ISO and not the ISO that contains the source files.
3. Upload the driver disk:

   ```
   xe [connection_parameters] update-upload file-name=
   <!--NeedCopy-->
   ```

   The UUID of the driver disk is returned when the upload completes.
4. Apply the driver disk:

   ```
   xe [connection_parameters] update-apply uuid=
   <!--NeedCopy-->
   ```

5. To complete the installation, restart the host. The driver does not take effect until after the host is restarted.

Upgrade from Citrix Hypervisor 8.2 Cumulative Update 1

April 25, 2024

By upgrading from an existing installation of Citrix Hypervisor 8.2 Cumulative Update 1 to XenServer 8, you can retain your existing VMs, SRs, and configuration.

Perform a rolling pool upgrade to keep all the services and resources offered by the pool available while upgrading all hosts in the pool. This upgrade method only takes one XenServer host offline at a time. Critical VMs are kept running during the process by live migrating the VMs to other hosts in the pool.

You can complete a rolling pool upgrade in one of the following ways:

- If you have a Premium Edition license, you can use the XenCenter **Rolling Pool Upgrade wizard**. This wizard organizes the upgrade path automatically and guides you through the upgrade procedure.

For more information, see Rolling pool upgrade by using XenCenter.

- You can use the xe CLI to perform a rolling pool upgrade manually by live migrating running VMs between XenServer hosts accordingly.

  For more information, see Rolling pool upgrade by using the xe CLI.

Can I upgrade?

Ensure that you are able to upgrade to XenServer 8:

- Are your hosts currently running Citrix Hypervisor 8.2 Cumulative Update 1?

  If not, you cannot upgrade directly to XenServer 8. Instead, perform a clean installation. For more information, see Install.

  If you are already using XenServer 8, do not attempt to update by using the installation ISO. Instead, apply updates to get your XenServer 8 pool to the latest level. For more information, see Apply updates.

- Are you using a supported partition layout?

  The legacy partition layout is no longer supported. If you use it, you might not be able to upgrade to XenServer 8. For more information, see Legacy partition layout.

- Are you using the Citrix Licensing Server virtual appliance?

  In previous releases, we supported the Linux-based License Server virtual appliance. This product is no longer supported. If you are using the License Server virtual appliance with an existing pool, migrate to the latest version of Citrix License Server for Windows before upgrading to XenServer 8. For more information, see Licensing.

- Is the key size of the server's identity certificate smaller than 2048 bytes?

  If your pool was first installed using XenServer 7.6 or earlier, it might still have certificates with a smaller key size than 2048 bytes. In this case, when you attempt to upgrade to XenServer 8, the upgrade wizard shows an error during prechecks. To proceed with the upgrade, you must reset the self-signed certificate on each affected server by running the following command:

  ```
  xe host-emergency-reset-server-certificate
  ```

  This command can interrupt ongoing operations in the pool.

- Is your hardware compatible with XenServer 8?

  Check that the hardware your pool is installed on is compatible with the version of XenServer you are about to upgrade to. For more information, see the Hardware Compatibility List (HCL).

- Are your VM operating systems supported by XenServer 8?

  Check that the operating systems of your VMs are supported by XenServer 8. If your VM operating system is not supported, upgrade your VM operating system to a supported version before upgrading XenServer. For more information, see Guest operating system support.

- Are you using XenServer to host your Citrix Virtual Apps and Desktops workloads?

  If you use your Citrix Virtual Apps and Desktops license to license your Citrix Hypervisor 8.2 Cumulative Update 1, this license no longer applies to XenServer 8. You must get a Xenserver Premium Edition license instead. For more information, see https://xenserver.com/buy.

  Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

  Apply your new licenses to your pool before beginning the upgrade.

  You can upgrade to XenServer 8 by using the methods described in this article. However, depending on your XenServer environment and your Citrix Virtual Apps and Desktops workload, there might be specific behaviors and requirements to consider that can optimize your XenServer upgrade process. For more information, see Upgrade scenarios for Citrix Virtual Apps and Desktops.

Before you start

Review the following information before starting your upgrade. Take the necessary steps to ensure that your upgrade process is successful.

Plan for the upgrade

1. Map your upgrade path carefully. Upgrading XenServer hosts, and particularly a pool of XenServer hosts, requires careful planning and attention to avoid losing any existing data.

   Note the following information when planning your upgrade:

   - You cannot migrate a VM from a newer version of XenServer to an older one.
   - Do not operate your pool in mixed-mode (with multiple versions of XenServer) for longer than is necessary. The pool operates in a degraded state during upgrade.
   - Key control operations are not available during the upgrade process. Do not attempt to perform any control operations.
   - Do not copy, shut down, or export VMs during the upgrade process.
   - Do not perform storage-related operations - such as adding, removing, or resizing virtual disks - during the upgrade process.

    - During the upgrade of the pool coordinator, the other hosts in the pool enter *emergency mode*.

2. Ensure that your servers are not over-provisioned: check that servers have sufficient memory to carry out the upgrade.

   Generally, if N equals the total number of servers in a pool, there must be sufficient memory across N-1 servers to run all live VMs in the pool. It is best to suspend any non-critical VMs during the upgrade process.

3. Ensure that your pool has shared storage to keep your VMs running during a rolling pool upgrade. If your pool does not have shared storage, you must suspend your VMs before upgrading because the VMs cannot be live migrated.

   Storage live migration is not supported with rolling pool upgrades.

4. If you use your Citrix Virtual Apps and Desktops license to license your Citrix Hypervisor 8.2 Cumulative Update 1, apply a Xenserver Premium Edition license to all hosts in the pool instead. For more information, see https://xenserver.com/buy.

5. If you perform a rolling pool upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8, you cannot use Workload Balancing 8.2.2 and earlier with your XenServer 8 pools. Update your Workload Balancing virtual appliance to version 8.3.0 before performing the rolling pool upgrade. You can download the latest version of the Workload Balancing virtual appliance from the XenServer Downloads page.

6. Note the following behaviors:

    - The upgrade must use the same boot mode as the initial install.
    - Boot-from-SAN settings are *not* inherited during the manual upgrade process. When upgrading using the ISO or PXE process, you must ensure that `multipathd` is correctly configured. For more information, see Boot from SAN.
    - When you upgrade XenServer, previously applied supplemental packs are removed and so they must be reapplied during or after the upgrade. However, the PVS-Accelerator supplemental pack is no longer required to be installed on XenServer 8. Its features are now included in the main product installation.

Prepare your pool

1. Take a backup of the state of your existing pool using the xe CLI command `xe pool-dump-database`.

   Taking a backup of the state ensures that you can revert a partially complete rolling upgrade to its original state without losing VM data.

2. Disable high availability.

Prepare your VMs

1. If you have Windows VMs running in your pool, take the following steps for each VM:
    - Ensure that the latest version of the XenServer VM Tools for Windows is installed.
    - Take a snapshot of the VM.
2. If you have Linux VMs running in your pool, ensure that the latest version of the XenServer VM Tools for Linux is installed.
3. If you have NVIDIA vGPU-enabled VMs running on your pool, complete the following steps to migrate the pool while these VMs are running:
    a) Ensure that the GPU you are using is supported on the version you plan to upgrade to.
    b) Identify a version of the NVIDIA GRID drivers that is available for both your current version of Citrix Hypervisor or XenServer and the version of XenServer you are upgrading to. If possible, choose the latest available drivers.
    c) Install the new GRID drivers on your XenServer hosts and the matching guest drivers on any of your vGPU-enabled VMs.
    d) Ensure that you also have the version of the GRID driver that matches the XenServer version that you are upgrading to. You are prompted to install these drivers as a supplemental pack as part of the rolling pool upgrade process.
4. Empty the CD/DVD drives of all VMs in the pool.

Get the required files

1. If you are using XenCenter to upgrade your hosts, download and install the latest version of XenCenter from the XenServer download site.

    For more information, see Install XenCenter.
2. Download the XenServer 8 installation ISO from the XenServer download site.
3. Prepare the installation media:
    - To upgrade your hosts from a bootable USB, use a tool like `rufus` or `diskpart` to create a bootable USB by using the XenServer 8 installation ISO. Ensure that the tool does not alter the contents of the ISO file.
    - To upgrade your hosts from a CD, burn the XenServer 8 installation ISO file to a CD.
    - To upgrade your hosts from virtual media, go to the virtual console of your system and mount the XenServer installation ISO file as virtual media.

- To upgrade from a network location:
  a) Set up a network-accessible TFTP server to boot the installer from.
  b) Set up a network location where you can access the installation ISO through HTTP, FTP, or NFS.
  c) Unpack the installation ISO onto the network location.

     If you are using IIS to host the installation media, ensure that double escaping is enabled on IIS before extracting the installation ISO on it.
  d) Make note of the information you need during the upgrade:
     - For HTTP or FTP, make note of the URL for your HTTP or FTP repository, and a user name and password, if appropriate.
     - For NFS, make note of the server and path of your NFS share.

  For more information, see Network boot.

After these prerequisite steps are completed, you can perform a rolling pool upgrade by one of the following methods:

- Rolling pool upgrade by using XenCenter
- Rolling pool upgrade by using the xe CLI

Rolling pool upgrade by using XenCenter

The **Rolling Pool Upgrade** wizard guides you through the upgrade procedure and organizes the upgrade path automatically. For pools, each of the servers in the pool is upgraded in turn, starting with the pool
coordinator. Before starting an upgrade, the wizard conducts a series of prechecks. These prechecks ensure certain pool-wide features, such as high availability, are temporarily disabled and that each server in the pool
is prepared for upgrade. Only one server is offline at a time. Any running VMs are automatically migrated off each server before the upgrade is installed on that server.

> **Note:**
>
> The XenCenter **Rolling Pool Upgrade wizard** is only available if you have a Premium Edition license.

If you have not yet installed XenCenter, download the latest version from the XenServer download site and complete the steps in Install XenCenter.

> **Note:**
>
> Use the latest XenCenter to upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8. Using an older version of XenCenter can result in a loss of connectivity.
>
> Download the latest XenCenter from the XenServer product downloads page.

The wizard can operate in manual or automatic mode:

- In manual mode, you must manually run the XenServer installer on each server in turn and follow the on-screen instructions on the serial console of the server. When the upgrade begins, XenCenter prompts you to insert the installation media or specify a network boot server for each server that you upgrade.
- In automatic mode, the wizard uses network installation files on an HTTP, NFS, or FTP server to upgrade each server in turn. This mode doesn't require you to insert installation media, manually reboot, or step through the installer on each server. If you perform a rolling pool upgrade in this manner, you must unpack the installation media onto your HTTP, NFS, or FTP server before starting the upgrade.

To upgrade XenServer hosts by using the XenCenter Rolling Pool Upgrade wizard:

1. On the XenCenter **Tools** menu, select **Rolling Pool Upgrade**.
2. Read the **Before You Start** information. Click **Next** to continue.
3. Select the pools and any individual hosts that you want to upgrade, and then click **Next**.
4. Choose one of the following modes:
    - **Automatic Mode** for an automated upgrade from network installation files on an HTTP, NFS, or FTP server.

      If you choose **Automatic Mode** and are using IIS to host the installation media, ensure that double escaping is enabled on IIS before extracting the installation ISO on it.
    - **Manual Mode** for a manual upgrade either from a USB/CD/DVD or by using network boot (using existing infrastructure).

      If you choose **Manual Mode**, you must run the XenServer installer on each host in turn. Follow the on-screen instructions on the serial console of the host. When the upgrade begins, XenCenter prompts you to insert the XenServer installation media or specify a network boot server for each host that you upgrade.
5. After you have selected your upgrade mode, click **Run Prechecks**.
6. Follow the recommendations to resolve any upgrade prechecks that have failed. If you want XenCenter to resolve all failed prechecks automatically, click **Resolve All**.

> **Note:**
>
> Some prechecks cannot be resolved automatically. For example, if your hosts are using a Citrix Virtual Apps and Desktops license, XenCenter shows that this license does not apply to XenServer 8 hosts. You can't upgrade until you get a XenServer Premium Edition license. For more information, see https://xenserver.com/buy.
>
> Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

7. When all prechecks have been resolved, click **Next** to continue.
8. Prepare the XenServer installation media.

   - If you chose **Automatic Mode**, enter the installation media details. Choose **HTTP**, **NFS**, or **FTP** and then specify the URL, user name, and password, as appropriate.

     > **Notes:**
     >
     > - If you choose FTP, ensure that you escape any leading slashes that are in the file path section of the URL.
     > - Enter the user name and password associated with your HTTP or FTP server, if you have configured security credentials. Do not enter the user name and password associated with your XenServer pool.
     > - XenServer supports FTP in passive mode only.

   - If you chose **Manual Mode**, note the upgrade plan and follow the instructions.

9. Click **Start Upgrade**.
10. When the upgrade begins, the **Rolling Pool Upgrade** wizard guides you through any actions you must take to upgrade each host. Follow the instructions until you have upgraded and updated all hosts in the pools.
11. If you have vGPU-enabled VMs, when you reach the step that gives you the option to supply a supplemental pack, upload the NVIDIA driver that matches the one on your vGPU-enabled VMs. Ensure you upload the version of the driver for the XenServer version you are upgrading to.
12. The **Rolling Pool Upgrade** wizard prints a summary when the upgrade is complete. Click **Finish** to close the wizard.

> **Note:**
>
> If the upgrade or the update process fails for any reason, the **Rolling Pool Upgrade** wizard halts

> the process. This allows you to fix the issue and resume the upgrade or update process by clicking the **Retry** button.

After the upgrade

After your pool is upgraded, we recommend that you complete the following tasks:

- Enable the certificate verification feature. For more information, see Certificate verification.
- Configure updates and apply the latest set. For more information, see Update your XenServer hosts.

After a rolling pool upgrade is complete, a VM might not be located on its home host. To relocate the VM, you can do one of the following actions:

- Live migrate the VM to its home host
- Shut down the VM and then start it on its home host

Rolling pool upgrade by using the xe CLI

Before performing a rolling pool upgrade by using the xe CLI, ensure that you have completed all the prerequisite steps in Before you start.

> **Important:**
>
> Ensure that you upgrade all servers in your pool. We strongly advise against running a mixed-mode pool (one with multiple XenServer versions) for longer than necessary, as the pool operates in a degraded state during upgrade.
>
> Key control operations are not available during the upgrade process. Do not attempt to perform any control operations. Though VMs continue to function as normal, VM actions other than migrate are not available (for example, shut down, copy and export). In particular, it is not safe to perform storage-related operations such as adding, removing, or resizing virtual disks.

To perform a rolling pool upgrade by using the xe CLI:

Start with the pool coordinator:

1. Disable the pool coordinator. This prevents any new VMs from starting on or being migrated to the specified host.

   ```
   xe host-disable host-selector=<host_selector_value>
   ```

2. Ensure that no VMs are running on the pool coordinator. Shut down, suspend, or migrate VMs to other hosts in the pool.

- To shut down a VM, use the following command:

  ```
  xe vm-shutdown
  ```

- To suspend a VM, use the following command:

  ```
  xe vm-suspend
  ```

- To migrate a specific VM, use the following command:

  ```
  xe vm-migrate
  ```

  Migrating specified VMs to specified hosts gives you full control over the distribution of migrated VMs to other hosts in the pool.

- To evacuate the host, use the following command:

  ```
  xe host-evacuate
  ```

  Evacuating all VMs from a host leaves the distribution of migrated VMs to XenServer.

3. Shut down the pool coordinator.

   ```
   xe host-shutdown
   ```

   > **Important:**
   >
   > You are unable to contact the pool coordinator until the upgrade of the pool coordinator is complete. Shutting down the pool coordinator causes the other hosts in the pool to enter *emergency mode*. Hosts can enter emergency mode when they in a pool whose pool coordinator has disappeared from the network and cannot be contacted after several attempts. VMs continue to run on hosts in emergency mode, but control operations are not available.

4. Boot the pool coordinator using the XenServer installation media and method of your choice (such as, USB or network).

5. Follow the XenServer installation procedure until the installer offers you the option to upgrade. Choose to upgrade.

   When your pool coordinator restarts, the other hosts in the pool leave emergency mode and normal service is restored after a few minutes.

6. Start or resume any shutdown or suspended VMs.

7. Migrate any VMs that you want back to the pool coordinator.

If anything interrupts the upgrade of the pool coordinator or if the upgrade fails for any reason, do not attempt to proceed with the upgrade. Reboot the pool coordinator and restore to a working version.

Repeat these steps for all the other hosts in the pool:

1. Select the next XenServer host in your upgrade path. Disable the host.

   ```
   xe host-disable host-selector=<host_selector_value>
   ```

2. Ensure that no VMs are running on the host. Shut down, suspend, or migrate VMs to other hosts in the pool.

   - To shut down a VM, use the following command:

     ```
     xe vm-shutdown
     ```

   - To suspend a VM, use the following command:

     ```
     xe vm-suspend
     ```

   - To migrate a specific VM, use the following command:

     ```
     xe vm-migrate
     ```

     Migrating specified VMs to specified hosts gives you full control over the distribution of migrated VMs to other hosts in the pool.

   - To evacuate the host, use the following command:

     ```
     xe host-evacuate
     ```

     Evacuating all VMs from a host leaves the distribution of migrated VMs to XenServer.

3. Shut down the host.

   ```
   xe host-shutdown
   ```

4. Boot the host using the XenServer installation media and method of your choice (such as, USB or network).

5. Follow the XenServer installation procedure until the installer offers you the option to upgrade. Choose to upgrade.

6. After the host upgrade is complete, start or resume any shutdown or suspended VMs.

7. Migrate any VMs that you want back to the host.

If the upgrade of a subordinate host fails or is interrupted, you do not have to revert. Run the command `xe host-forget` in the pool to forget that host. Reinstall XenServer on the host, and then join it, as a new host, to the pool using the command `xe pool-join`.

After the upgrade

After your pool is upgraded, we recommend that you complete the following tasks:

- Enable the certificate verification feature. For more information, see Certificate verification.
- Configure updates and apply the latest set. For more information, see Update your XenServer hosts.

After a rolling pool upgrade is complete, a VM might not be located on its home host. To relocate the VM, you can do one of the following actions:

- Live migrate the VM to its home host
- Shut down the VM and then start it on its home host

Other scenarios

Legacy partition layout

The legacy partition layout is no longer supported. If you use it, you might not be able to upgrade to XenServer 8 and must instead perform a fresh installation.

XenServer 6.5 and earlier uses a 4 GB control domain (dom0) partition for all dom0 functions, including swap and logging. This partition configuration is referred to as the legacy partition layout. Later releases of XenServer and Citrix Hypervisor introduced a partition layout that increased the control domain partition to 18 GB and included a separate logging partition. In XenServer 8, only the newer partition layout is supported.

How do I know what partition layout my server uses? You might have the legacy partition layout on your XenServer hosts in the following cases:

- You initially installed your XenServer host with XenServer 5.6 Service Pack 2 or earlier and have since upgraded to later supported versions.
- You are using old hardware that has less than 46 GB of primary disk space.
- Your hardware requires that a utility partition is present.

To find out how many partitions your XenServer host has, run the following command in the server console:

```
fdisk -l
```

- If the command lists 6 partitions, you are using the new partition layout and can upgrade to XenServer 8.
- If the command lists 3 or 4 partitions, you are using the legacy partition layout.

What can I do next? If you are using the new partition layout, you can upgrade to XenServer 8.

If you are using the legacy partition layout:

- If you have less than 46 GB of primary disk space or your hardware requires that a utility partition is present, you cannot install or upgrade to XenServer 8.
- If your disk is GPT and the local SR is empty with at least 38 GB free, you can switch from the legacy partition layout to the new partition layout during upgrade. You must use XenCenter to attempt the upgrade on a server with the legacy partition layout. For more information, see Rolling pool upgrade by using XenCenter.
- For other hardware, you can complete a fresh installation of XenServer 8. For more information, see Install.

Citrix Virtual Apps and Desktops environments

If you are using XenServer to host your Citrix Virtual Apps and Desktops workloads, see Upgrade scenarios for Citrix Virtual Apps and Desktops.

Update your XenServer hosts

March 28, 2024

With XenServer 8, new features and bug fixes are frequently pushed to the content delivery network (CDN) as available updates for your XenServer hosts and pools, allowing you to benefit from a more efficient release process that delivers new content to you at a faster cadence than was previously possible.

To ensure that you are always on the latest and greatest update, there is no picking and choosing - when you apply updates to your pool, it is updated to the latest, fully tested state. Configure your pool to automatically synchronize with an update channel. This action downloads all available updates to the pool coordinator. You can then apply all downloaded updates by using XenCenter or the xe CLI.

> **Note:**
>
> If you used XenServer 8 during its preview period, you do not need to reinstall or upgrade your hosts to move to the GA version. Apply the latest updates through XenCenter to be supported in production.

Lifecycle

During its lifecycle, XenServer 8 provides a stream of frequent and easy-to-apply updates, which enable you to consume new features and bug fixes at the earliest possible juncture. You must apply all available updates periodically. As a result, the behavior and feature set in XenServer 8 can change.

For more information, see XenServer lifecycle.

Update process

The XenServer 8 release stream and the content delivery network (CDN) work together to enable you to apply frequent updates to your XenServer hosts and pools from XenCenter.

1. We make frequent updates available for XenServer 8 in our secure CDN.
2. In XenCenter, see when updates are available for your pool.
3. Using XenCenter, initiate the process of applying updates to your XenServer pool.

Update channels

The XenServer 8 release stream consists of two phases, also referred to as update channels:

- Early Access
- Normal

To receive frequent updates, configure your XenServer pool to subscribe to one of those update channels.

1. When updates are first pushed to our CDN, they enter the Early Access update channel.

   Early Access is perfect for test environments, allowing you to get the latest updates when they're released to the public. By opting into receiving updates early, you have the opportunity to trial them before they're made available to the Normal update channel.

   > **Note:**
   >
   > Early Access is supported for production use. However, we do not recommend it for critical production environments.

2. These updates then flow sequentially into Normal, the next update channel.

   Unless we have delayed this progression, you can expect to see Early Access updates become available in Normal on a regular cadence. Normal is recommended for production environments.

Occasionally, you might see that updates become available to both your Early Access and Normal pools at the same time. These updates enable us to deliver security patches and critical fixes to all update channels immediately.

Get started with updates

For information on how to configure and apply updates for your XenServer hosts by using XenCenter, see Apply updates by using XenCenter. Alternatively, you can use the xe CLI to apply updates to your XenServer hosts. For more information, see Apply updates by using the xe CLI.

> **Important:**
>
> We do not support the direct usage or modification of the underlying update components in dom0. You can only use XenCenter or the xe CLI to configure and apply updates.

Apply updates by using XenCenter

March 26, 2024

Apply updates to your XenServer 8 hosts and pools by using the latest version of XenCenter. The latest version of XenCenter is provided on the XenServer product downloads page.

To provide update notifications, XenCenter requires internet access. If your XenCenter is behind a firewall, ensure that it has access to the `updates.ops.xenserver.com` domain. To receive the updates, your XenServer hosts require internet access. If your hosts are behind a firewall, ensure that they have access to subdomains of `ops.xenserver.com`. For more information, see Connectivity requirements.

Complete the following steps to be able to update your XenServer pools:

1. Install the latest version of XenCenter.
2. Install or upgrade to XenServer 8.
3. Configure updates for your pool.
4. View available updates for your pool.
5. Apply updates to your pool.
6. Carry out the update tasks.

Configure updates for your pool

Before being able to apply updates to your XenServer hosts and pools, you must configure host updates by subscribing your pool or host to an update channel. These channels control how soon you can access updates that are made available in the content delivery network (CDN).

The two update channels are:

- Early Access - perfect for test environments
- Normal - recommended for production environments

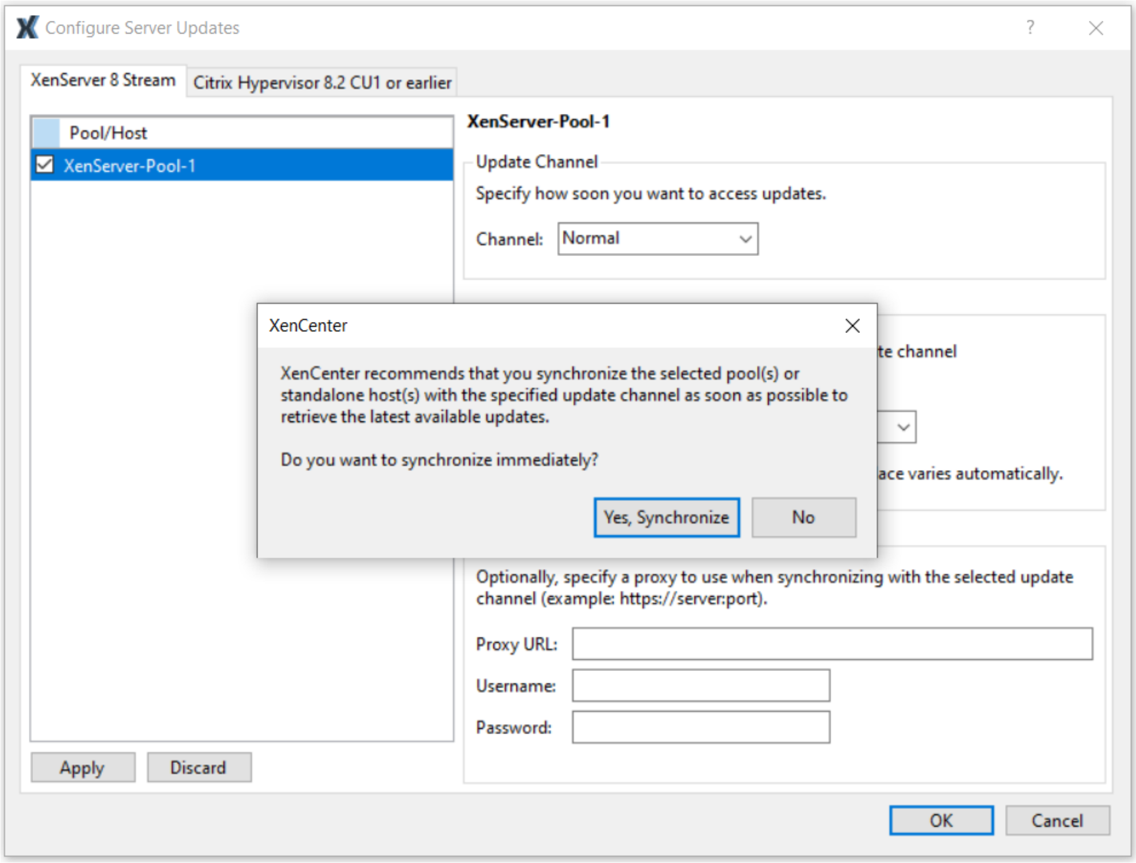
After subscribing your pool to one of the update channels, your pool regularly and automatically synchronizes with the update channel. This action downloads all available updates to the pool coordinator. You can then apply all of the downloaded updates by using XenCenter.

1. In XenCenter, on the **Tools** menu, select **Configure Updates**. Alternatively, go to the **Updates** section under your pool's **General** tab and select **Configure Updates** or right-click on your pool and select **Updates** > **Configure Updates**. The **Configure Server Updates** window opens.
2. On the **XenServer 8** tab, select the pools or hosts that you want to configure.
3. Under **Update Channel**, specify how soon you want to access updates. Your pool or host can be subscribed to one of the following update channels:
   - **Early Access**
   - **Normal**
4. Under **Synchronization Schedule**, select how often you want your XenServer pool to synchronize with the update channel. This can be daily, or weekly on a certain day of the week.

   > **Note:**
   >
   > After synchronizing, apply the updates to your pool as soon as possible to benefit from the latest updates.
   >
   > If you designate a new pool coordinator after synchronizing but before applying updates to the hosts in the pool, you must synchronize again with the new pool coordinator before you can update the pool.
   >
   > Do not synchronize your XenServer pool while the pool is in the process of being updated.

5. (Optional) Under **Proxy Server**, specify a proxy to use when synchronizing with the update channel. This proxy server is used for communication between the host and the public CDN.
6. Click **Apply** to apply the configuration changes to your XenServer pool and then repeat the above steps to configure updates for the rest of your XenServer pools.

7. If you are happy with the configuration changes to your pools, click **OK** to save your changes and close the **Configure Server Updates** window. When you first set up your host or pool with an update channel (or if you later change your host or pool to synchronize with a different update channel), you are asked if you want to synchronize your host or pool with the update channel immediately. In the dialog box that opens, select **Yes, Synchronize** if you want to synchronize your host or pool with the update channel immediately.


Configure Server Updates
XenServer 8 Stream
Citrix Hypervisor 8.2 CU1 or earlier
Pool/Host
XenServer-Pool-1
XenServer-Pool-1
Update Channel
Specify how soon you want to access updates.
Channel: Normal
te channel
ace varies automatically.
XenCenter
Yes, Synchronize
No
Optionally, specify a proxy to use when synchronizing with the selected update channel (example: https://server:port).
Proxy URL:
Username:
Password:
Apply
Discard
OK
Cancel

XenCenter recommends that you synchronize the selected pool(s) or standalone host(s) with the specified update channel as soon as possible to retrieve the latest available updates.

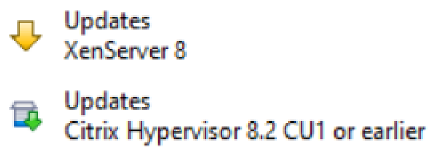
Do you want to synchronize immediately?

8. As soon as your pool synchronizes with the update channel, apply the downloaded updates to your pool by using the **Install Updates** wizard. For more information, see Apply updates to your pool.

After configuring your XenServer pool, you can find information about the update channel that your pool is subscribed to and the last time your pool synchronized with the update channel in the **Updates** section under your pool's **General** tab in XenCenter. You can also find information about the last time your host was updated in the **Updates** section under your host's **General** tab.

View available updates for your pool

XenCenter issues notifications about available updates for your hosts and pools under the **Updates** tabs in the **Notifications** view. The **Updates** tabs are split into XenServer 8 updates and Citrix Hypervisor updates.



The XenServer 8 **Updates** tab refreshes when your XenServer 8 hosts and pools synchronize with the update channel. The frequency of this refresh depends on the synchronization schedule that you have set up for your pool (either daily, or weekly on a certain day of the week).

To see the latest available updates for your pool, synchronize your XenServer pool with the update channel. You can do this from the following places:

- In the **By Server** view of the **Updates** tab, you can choose to **Synchronize All** to synchronize all pools managed by XenCenter or **Synchronize Selected** to synchronize the selected pools.
- Alternatively, go to the **Updates** section under your pool's **General** tab and select **Synchronize Now** or right-click on your pool and select **Updates** > **Synchronize Now**.

You can then review all updates available for your XenServer 8 pools. The types of updates are:

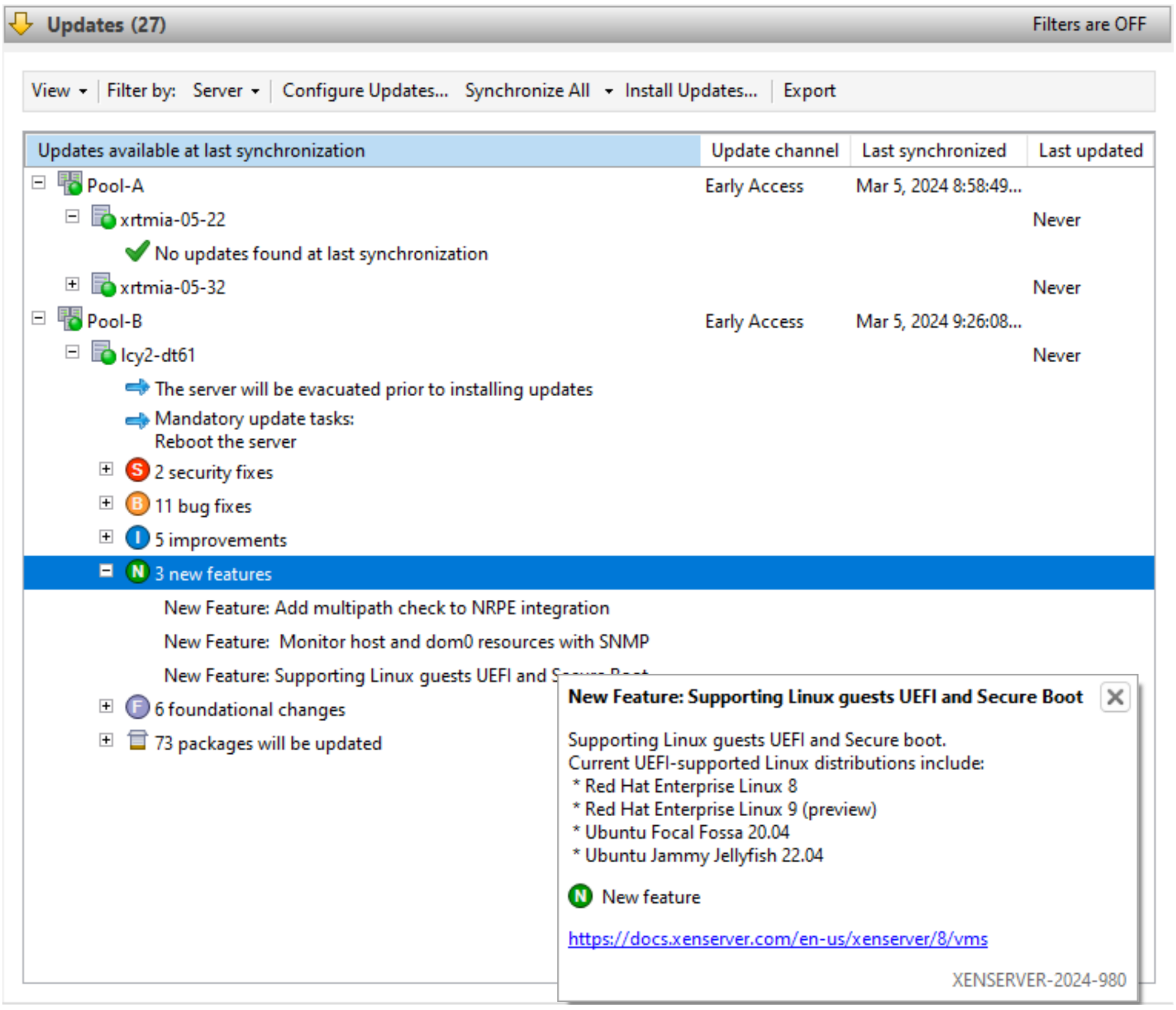
- **Security fixes**
- **Bug fixes**
- **Improvements**
- **New features**
- **Preview features**
- **Foundational changes**

> **Note:**
>
> Foundational changes are non-customer visible groundwork changes to maintain and improve the product.

In the main panel, use the **View** option to choose whether the updates are displayed **By Server** or **By Update**.

By server

Updates are grouped by host and type of update.


Updates (27)

Filters are OFF

Updates available at last synchronization	Update channel	Last synchronized	Last updated
Pool-A	Early Access	Mar 5, 2024 8:58:49...	
xrtmia-05-22			Never
No updates found at last synchronization			
xrtmia-05-32			Never
Pool-B	Early Access	Mar 5, 2024 9:26:08...	
lcy2-dt61			Never
The server will be evacuated prior to installing updates			
Mandatory update tasks: Reboot the server			
2 security fixes			
11 bug fixes			
5 improvements			
3 new features			
New Feature: Add multipath check to NRPE integration			
New Feature: Monitor host and dom0 resources with SNMP			
New Feature: Supporting Linux guests UEFI and Secure Boot			
6 foundational changes			
73 packages will be updated			

New Feature: Supporting Linux guests UEFI and Secure Boot

Supporting Linux guests UEFI and Secure boot.
Current UEFI-supported Linux distributions include:
* Red Hat Enterprise Linux 8
* Red Hat Enterprise Linux 9 (preview)
* Ubuntu Focal Fossa 20.04
* Ubuntu Jammy Jellyfish 22.04

New feature

https://docs.xenserver.com/en-us/xenserver/8/vms

XENSERVER-2024-980

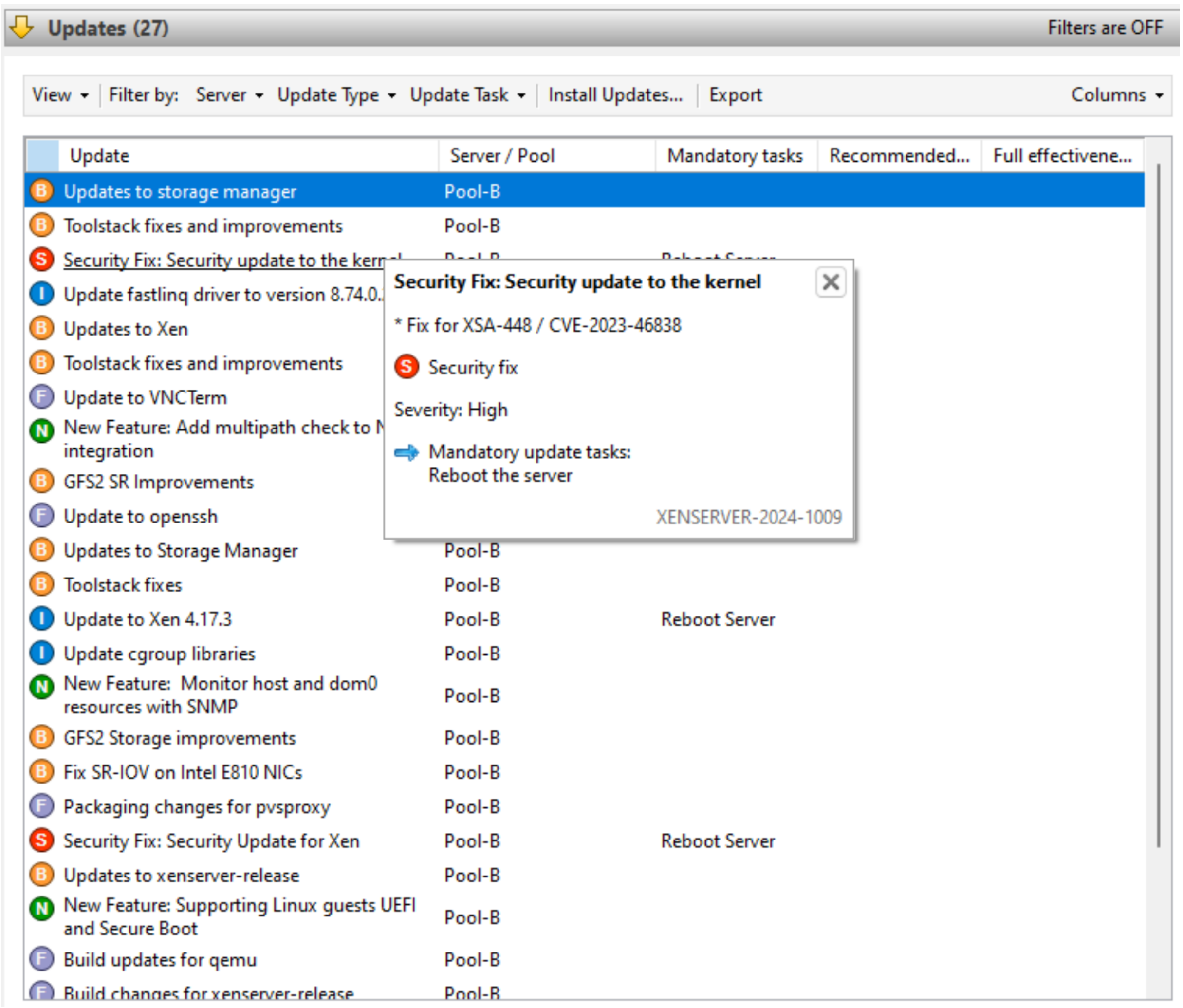
You can filter the update information by server. Select an update and hover over it to view detailed information about the update.

To view this information about your available updates offline, select **Export All** to export the information as a `.md` file. The contents of the `.md` file are grouped by pool and then host. For each host, the file lists the following information:

- Any update tasks for this host or its VMs. For more information about mandatory, recommended, and full effectiveness tasks, see Update tasks.
- Updates grouped by update type
  - The update name
  - A description of the update
- A list of the RPMs to be updated on the host

By update

All updates are listed chronologically in order of release.



Updates (27) Filters are OFF

View | Filter by: Server | Update Type | Update Task | Install Updates... | Export | Columns

	Update	Server / Pool	Mandatory tasks	Recommended...	Full effectivene...
B	Updates to storage manager	Pool-B			
B	Toolstack fixes and improvements	Pool-B			
S	Security Fix: Security update to the kern	Pool-B	Reboot Server		
I	Update fastlinq driver to version 8.74.0.				
B	Updates to Xen				
B	Toolstack fixes and improvements				
F	Update to VNCTerm				
N	New Feature: Add multipath check to N integration				
B	GFS2 SR Improvements				
F	Update to openssh				
B	Updates to Storage Manager	Pool-B			
B	Toolstack fixes	Pool-B			
I	Update to Xen 4.17.3	Pool-B	Reboot Server		
I	Update cgroup libraries	Pool-B			
N	New Feature: Monitor host and dom0 resources with SNMP	Pool-B			
B	GFS2 Storage improvements	Pool-B			
B	Fix SR-IOV on Intel E810 NICs	Pool-B			
F	Packaging changes for pvsproxy	Pool-B			
S	Security Fix: Security Update for Xen	Pool-B	Reboot Server		
B	Updates to xenserver-release	Pool-B			
N	New Feature: Supporting Linux guests UEFI and Secure Boot	Pool-B			
F	Build updates for qemu	Pool-B			
F	Build changes for xenserver-release	Pool-B			

Security Fix: Security update to the kernel

* Fix for XSA-448 / CVE-2023-46838

S Security fix

Severity: High

Mandatory update tasks:
Reboot the server

XENSERVER-2024-1009

You can filter the update information by the server it can be applied to, by the update type, and by any update tasks that apply to it. Select an update and hover over it to view detailed information about the update.

To view this information about your available updates offline, select **Export All** to export the information as a `.csv` file. The `.csv` file contains the following information:

- The update type
- The update name
- The servers that this update can be applied to
- The mandatory, recommended, and full effectiveness tasks

For more information about mandatory, recommended, and full effectiveness tasks, see Update tasks.

- To apply the updates to your hosts or pools, select **Install Updates** to open the **Install Updates** wizard. For more information, see the following section Apply updates to your pool.

Apply updates to your pool

The update installation mechanism in XenCenter applies the updates to your hosts and pools using the **Install Updates** wizard. During the process, XenCenter automatically works out the least impactful action required after applying all available updates. The **Install Updates** wizard automatically performs these steps:

1. If required, it migrates VMs off each host.
2. If required, it places the host in Maintenance mode.
3. It applies the updates.
4. If required, it runs any necessary update tasks such as rebooting the host, restarting the toolstack, or rebooting the VMs.
5. It migrates the VMs back to the updated host.

Any actions that are taken at the pre-check stage to enable the updates to be applied, such as turning off high availability, are reverted.

Before you start

Before you apply an update to your servers, pay careful attention to the following:

- Back up all your servers
- Ensure that High Availability (HA) is not enabled on any pool where you intend to apply updates.
- Ensure that you are logged in to XenCenter as a Pool Administrator or Pool Operator, or using a local root account.

Install updates

The following section provides step-by-step instructions on applying updates using the **Install Updates** wizard:

1. From the XenCenter menu, select **Tools** and then **Install Updates**.
2. In the **Install Updates** wizard, select **XenServer 8** and review the information on the **Before You Start** page. Click **Next** to continue.
3. Select **Automated updates**. Click **Next**.
4. Select your XenServer pools or hosts that you want to update. Click **Next** to progress to the next wizard page and begin pre-checks.

5. The wizard performs several pre-checks to verify that the updates can be applied on your host or pool. For example, you must have synchronized your host or pool with the update channel within the past week.

   Follow the on-screen recommendations to resolve any pre-checks that have failed. If you prefer XenCenter to automatically resolve all failed pre-checks, select **Resolve All**. When the prechecks have been resolved, select **Next** to continue.

   > **Notes:**
   >
   > - If the update process cannot complete for any reason, XenCenter halts the process. This halt allows you to fix the issue and resume the update process by clicking the **Retry** button.
   > - If you select **Cancel** at this stage, the **Install Updates** wizard reverts any changes.

6. After updates are applied, some update tasks (such as rebooting your hosts) might be required. On the **Update Mode** page, select the level of update tasks that you want XenCenter to automatically carry out (such as restarting your hosts) after applying updates to your pool. By default, XenCenter chooses the recommended level of update tasks. Mandatory tasks cannot be deselected and XenCenter performs these tasks automatically.

   The mandatory, recommended, and full effectiveness update tasks are listed under **Tasks**. If there are no update tasks required, the page displays a note saying **No action required**. For more information about the different types of update tasks and the guidance levels provided by XenCenter, see Update tasks.

7. Click **Install updates** for XenCenter to begin installing updates for your host or pool.

8. The **Install Updates** wizard shows the progress of the update, displaying the major operations that XenCenter performs while updating each host in the pool. Click **Finish** to complete the updates and close the **Install Updates** wizard.

Update tasks

Some tasks (such as evacuating or rebooting your hosts) might be required before and after applying updates to your pool. Sometimes, there are no update tasks required.

Guidance categories

XenServer tries to minimize the disruption to your VMs that these update tasks might cause by categorizing the tasks into **Mandatory**, **Recommended**, **Full-effectiveness**, and **Live patch**. These categorizations enable you to judge whether an update task that might cause downtime or minor disruption for your hosts or VMs is necessary for your environment and risk profile.

Updates can have tasks listed in more than one of these categories. For example, an update might require that you restart the host to get the full-effectiveness of the update, but recommend restarting the toolstack to get most of the benefit of the update with less potential disruption to the pool.

During the update process, you can choose to carry out one of the following three levels of tasks:

1. Mandatory
2. Mandatory + Recommended
3. Mandatory + Recommended + Full-effectiveness

Mandatory Mandatory tasks *must* be performed after an update otherwise the system might fail at runtime. These actions are required to enable critical fixes and ensure that your environment is secure and stable. When you apply updates, XenCenter performs these tasks. You cannot opt out of mandatory tasks.

Recommended Recommended tasks are the tasks that we recommend you perform to get the benefit of the majority of features and fixes delivered in the updates. When you apply updates, these tasks are selected by default in XenCenter, but you can opt out of performing them. If you choose not to perform these tasks now, they are listed in the pending tasks for the applicable pool, host, or VM.

Why perform the recommended tasks:

- These tasks are the ones that ensure a secure, stable XenServer environment.

Why opt out of the recommended tasks:

- After reviewing the detailed information for the updates, you judge that the risk of not fully applying these updates now is acceptable.
- The recommended tasks cause unwanted disruption to your VMs now.

Full-effectiveness Full effectiveness tasks are required to get the benefit of the related update. The updates that have full-effectiveness tasks associated with them are usually relevant only to users on certain hardware or using specific features.

Review the update information to understand whether these tasks are required for your environment. When you apply updates, these tasks are not selected by default in XenCenter, but you can choose to perform them during the update if you believe the update applies to your environment or configuration. If you choose not to perform these tasks now, they are listed in the pending tasks for the applicable pool, host, or VM.

Why perform the full-effectiveness tasks:

- The updates that have full-effectiveness tasks are relevant to your hardware, environment, or configuration.

Why remain opted out of the full-effectiveness tasks:

- The updates that have full-effectiveness guidance are not relevant to your hardware, environment, or configuration.
- The full-effectiveness tasks cause unwanted disruption to your VMs now.
- You do not need the benefits of these updates right now.

If the full-effectiveness tasks apply to your environment, but you have opted to defer them, plan to complete these tasks during a suitable maintenance window to maintain the stability of your environment.

Live patches Updates to certain components can include a live patch. Whether a live patch can be applied to your hosts depends on the version of the component that was installed when the hosts were last rebooted. If an update can be applied as a live patch to your hosts, the live patch guidance replaces the recommended guidance.

> **Example:**
>
> You have two pools. Pool A is updated to a recent level. Pool B has not been updated for some time. We release a new update that has the recommended update task "Restart host"and the live patch update task "Restart toolstack".
>
> In pool A, the live patch can be applied to these more up-to-date hosts. XenCenter recommended guidance shows "Restart toolstack". The less disruptive task from the live patch guidance overrides the recommended guidance.
>
> In pool B, the live patch cannot be applied to the hosts as they are at an older level. XenCenter recommended guidance shows "Restart host". The recommended guidance remains applicable. The live patch guidance is irrelevant in this case.

Sometimes only some of the fixes in an update are enabled when the update is applied as a live patch. Review the update details to understand whether you need all fixes in the update or just those fixes enabled by the live patch. You can then use this information to choose whether to perform the recommended tasks. For more information, see View available updates for your pool.

Update tasks

One or more of the following tasks might be required when applying an update. Any type of update task can be listed in any guidance category.

Update tasks for your host You only ever carry this task out *before* applying updates and you sometimes carry it out as part of the 'Reboot host'task:

- **Evacuate server**: All VMs must be migrated off the XenServer host or shutdown before applying the update. To complete this task, XenCenter migrates any VMs off the host. While this task is in progress, the XenServer pool is operating at reduced capacity as one host is temporarily unavailable to run VMs.

The following tasks require actions on the updated host:

- **Reboot server**: The XenServer host must be restarted. To complete this task, XenCenter migrates any VMs off the host and restarts the host. While this task is in progress, the XenServer pool is operating at reduced capacity as one host is temporarily unavailable to run VMs.
- **Restart toolstack**: The toolstack on the host must be restarted. When XenCenter restarts the toolstack on the pool coordinator, XenCenter loses connection to the pool and automatically attempts to reconnect. On other pool members, there is no visible effect.

Update tasks for your VM Some updates provide new features for your VMs. These updates might require the following tasks on your VMs:

- **Reboot VM**: The VM must be restarted. In XenCenter, while the VM is restarting it shows the red stop icon (square on red). When the task is complete, it shows the green play icon. During this time, the VM is unavailable to the end user.
- **Restart device model**: The device model for VMs on the updated host must be restarted. In XenCenter, while the device model is restarting the VM shows a yellow warning triangle. When the task is complete, it shows the green play icon. During this time you can't stop, start, or migrate the VM. The end user of the VM might see a slight pause and resume in their session.

  For the restart device model action to be supported on a Windows VM, the VM must have the XenServer VM Tools for Windows installed.

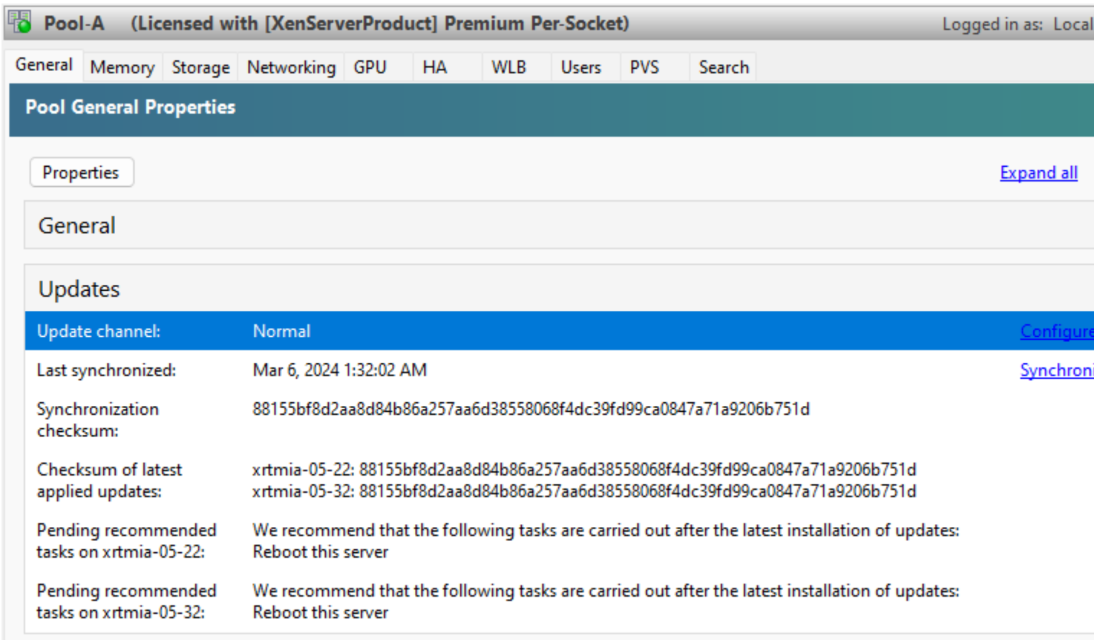
Review update tasks before applying updates

These tasks are listed on the XenServer 8 **Updates** tab in the **Notifications** view. For more information, see View available updates for your pool.

View pending tasks

If you choose not to perform all tasks during an update, the pending tasks for each pool, host, or VM is shown in the XenCenter **Infrastructure** view.

In the **General** tab for the pool, host, or VM, see the **Updates** section.


Pool-A (Licensed with [XenServerProduct] Premium Per-Socket)

Logged in as: Local

Pool General Properties

General

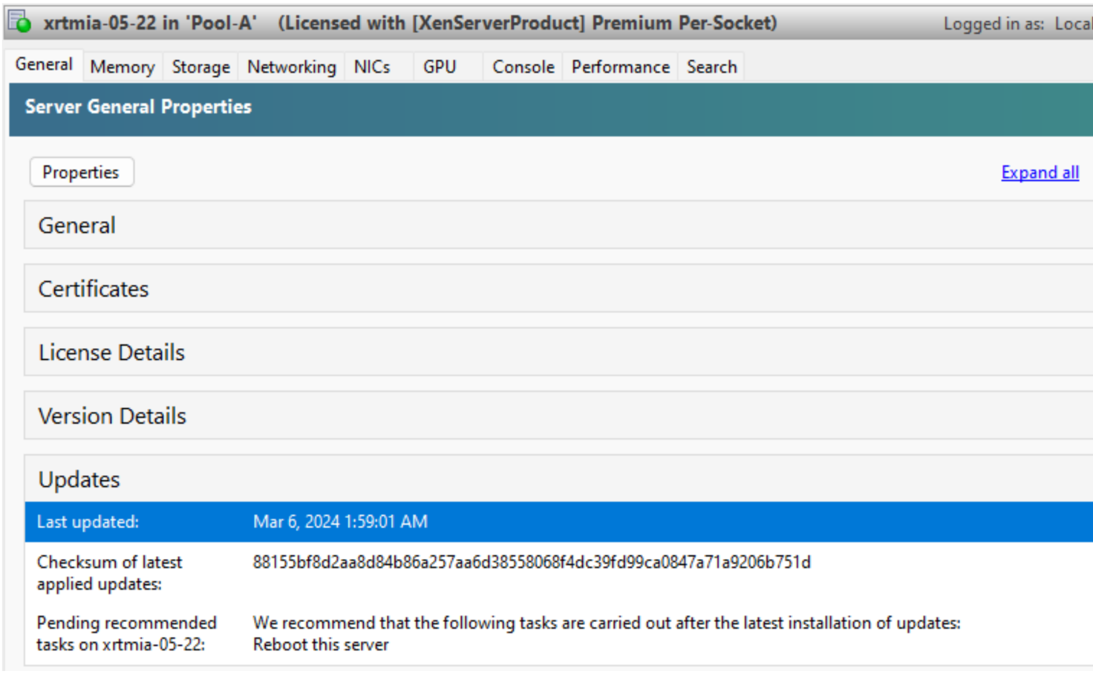
Updates

Update channel:	Normal	Configure
Last synchronized:	Mar 6, 2024 1:32:02 AM	Synchroni
Synchronization checksum:	88155bf8d2aa8d84b86a257aa6d38558068f4dc39fd99ca0847a71a9206b751d	
Checksum of latest applied updates:	xrtmia-05-22: 88155bf8d2aa8d84b86a257aa6d38558068f4dc39fd99ca0847a71a9206b751d xrtmia-05-32: 88155bf8d2aa8d84b86a257aa6d38558068f4dc39fd99ca0847a71a9206b751d	
Pending recommended tasks on xrtmia-05-22:	We recommend that the following tasks are carried out after the latest installation of updates: Reboot this server	
Pending recommended tasks on xrtmia-05-32:	We recommend that the following tasks are carried out after the latest installation of updates: Reboot this server	

Pool pending tasks

This section shows the pending tasks for all hosts in the pool.

It also shows a checksum that indicates the level that the current pool coordinator has synchronized to and checksums for each host that indicate the level of the updates installed. These checksums can provide useful information if you need to contact Technical Support.


xrtmia-05-22 in 'Pool-A' (Licensed with [XenServerProduct] Premium Per-Socket)

Logged in as: Local

Server General Properties

General

Certificates

License Details

Version Details

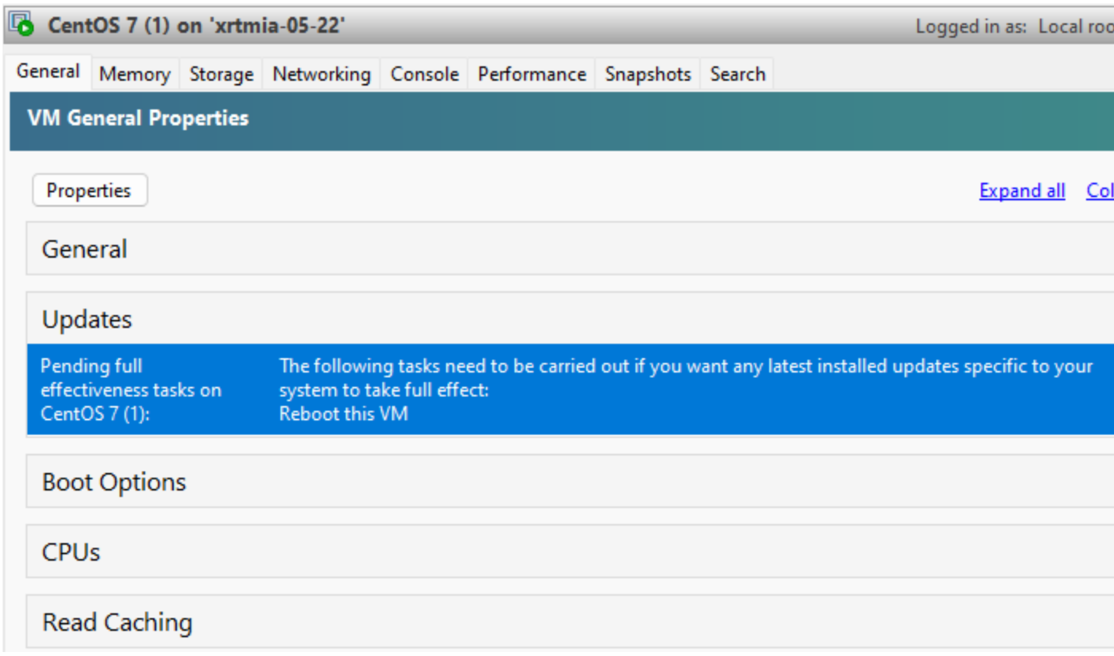
Updates

Last updated:	Mar 6, 2024 1:59:01 AM
Checksum of latest applied updates:	88155bf8d2aa8d84b86a257aa6d38558068f4dc39fd99ca0847a71a9206b751d
Pending recommended tasks on xrtmia-05-22:	We recommend that the following tasks are carried out after the latest installation of updates: Reboot this server

Host pending tasks

This section shows the pending tasks for the XenServer host.

It also shows a checksum that indicates the level of the updates installed. This checksum can provide useful information if you need to contact Technical Support.


CentOS 7 (1) on 'xrtmia-05-22'

Logged in as: Local roo

Pending full effectiveness tasks on CentOS 7 (1): The following tasks need to be carried out if you want any latest installed updates specific to your system to take full effect: Reboot this VM

VM pending tasks

This section shows the pending tasks for a VM.

Apply updates by using the xe CLI

April 24, 2024

Apply updates to your XenServer 8 hosts and pools by using the xe CLI.

To receive updates, your XenServer hosts require internet access. If your hosts are behind a firewall, ensure that they have access to subdomains of `ops.xenserver.com`. For more information, see Connectivity requirements.

Complete the following steps to be able to update your XenServer pools:

1. Install or upgrade to XenServer 8.
2. Configure updates for your pool.
3. Synchronize new updates for your pool.
4. Understand the guidance categories and update tasks.
5. View available updates for your pool.
6. Apply updates to your pool.

Configure updates for your pool

Before being able to apply updates to your XenServer hosts and pools, you must configure host updates by subscribing your pool or host to an update channel. These channels control how soon you can access updates that are made available in the content delivery network (CDN).

The two update channels are:

- **Early Access** - perfect for test environments
- **Normal** - recommended for production environments

After subscribing your pool to one of the update channels, your pool regularly and automatically synchronizes with the update channel. Alternatively, manually synchronize your pool with the update channel. This synchronization action downloads all available updates to the pool coordinator. You can then apply the downloaded updates by using the xe CLI.

1. Create and enable the Early Access update channel for your pool:

   ```
   pool_uuid=$(xe pool-list --minimal)

   base_binary_url="https://repo.ops.xenserver.com/xs8/base"
   base_source_url="https://repo-src.ops.xenserver.com/xs8/base"
   base_repo_uuid=$(xe repository-introduce name-label=base_repo name
       -description=Base binary-url=<base_binary_url> source-url=<
       base_source_url> update=false)

   update_binary_url="https://repo.ops.xenserver.com/xs8/earlyaccess"
   update_source_url="https://repo-src.ops.xenserver.com/xs8/
       earlyaccess"
   update_repo_uuid=$(xe repository-introduce name-label=
       early_access_repo name-description="Early Access" binary-url=<
       update_binary_url> source-url=<update_source_url> update=true)

   xe pool-param-set uuid=<pool_uuid> repositories=<base_repo_uuid>,<
       update_repo_uuid>
   <!--NeedCopy-->
   ```

   Alternatively, create and enable the Normal update channel for your pool:

   ```
   pool_uuid=$(xe pool-list --minimal)

   base_binary_url="https://repo.ops.xenserver.com/xs8/base"
   base_source_url="https://repo-src.ops.xenserver.com/xs8/base"
   base_repo_uuid=$(xe repository-introduce name-label=base_repo name
       -description=Base binary-url=<base_binary_url> source-url=<
       base_source_url> update=false)

   update_binary_url="https://repo.ops.xenserver.com/xs8/normal"
   update_source_url="https://repo-src.ops.xenserver.com/xs8/normal"
   update_repo_uuid=$(xe repository-introduce name-label=normal name-
       description="Normal" binary-url=<update_binary_url> source-url
   ```

```
    =<update_source_url> update=true)

xe pool-param-set uuid=<pool_uuid> repositories=<base_repo_uuid>,<
    update_repo_uuid>
<!--NeedCopy-->
```

2. Retrieve a list of the currently enabled repository UUIDs:

```
pool_uuid=$(xe pool-list --minimal)
xe pool-param-get uuid=<pool_uuid> param-name=repositories
<!--NeedCopy-->
```

3. Using the repository UUID, view more details about a particular repository:

```
xe repository-param-list uuid=<UUID>
<!--NeedCopy-->
```

4. (Optional) Configure and enable an HTTP connect proxy server which is used for communication between the host and the public CDN that hosts the repositories:

```
xe pool-configure-repository-proxy proxy-url=<http://proxy.example
    .com> proxy-username=<proxy-user> proxy-password=<proxy-
    password>
<!--NeedCopy-->
```

Disable the proxy server configuration:

```
xe pool-disable-repository-proxy
<!--NeedCopy-->
```

View the proxy server configuration:

```
pool_uuid=$(xe pool-list --minimal)
xe pool-param-get uuid=<pool_uuid> param-name=repository-proxy-url
xe pool-param-get uuid=<pool_uuid> param-name=repository-proxy-
    username
<!--NeedCopy-->
```

Synchronize new updates for your pool

Enable your pool to automatically synchronize with the update channel by configuring a synchronization schedule. You can schedule a synchronization to take place daily or weekly on a certain day of the week. Synchronizing your pool with the update channel downloads all available updates to the pool coordinator and you can then apply all of the downloaded updates to your pool.

1. Set your pool to synchronize daily:

```
xe pool-configure-update-sync update-sync-frequency=daily update-
    sync-day=0
```

```
xe pool-set-update-sync-enabled value=true
<!--NeedCopy-->
```

Alternatively, set your pool to synchronize weekly:

```
xe pool-configure-update-sync update-sync-frequency=weekly update-
    sync-day=1 (# 0 is Sunday, 1 is Monday, etc)
xe pool-set-update-sync-enabled value=true
<!--NeedCopy-->
```

2. View your synchronization configuration:

```
pool_uuid=$(xe pool-list --minimal)
xe pool-param-get uuid=<pool_uuid> param-name=update-sync-
    frequency
xe pool-param-get uuid=<pool_uuid> param-name=update-sync-day
xe pool-param-get uuid=<pool_uuid> param-name=update-sync-enabled
<!--NeedCopy-->
```

3. Get the timestamp of your pool's last successful synchronization with the update channel:

```
pool_uuid=$(xe pool-list --minimal)
xe pool-param-get param-name=last-update-sync uuid=<pool_uuid>
<!--NeedCopy-->
```

Alternatively, you can manually synchronize your XenServer pool with the update channel:

```
pool_uuid=$(xe pool-list --minimal)
update_checksum=$(xe pool-sync-updates uuid=<pool_uuid> --minimal)
<!--NeedCopy-->
```

`update_checksum` is a unique identifier that indicates the level of the updates installed. It changes whenever new updates are made available in the public CDN and is later used when applying updates to your pool to ensure that you are always applying the latest available updates. `update_checksum` can also provide useful information if you need to contact Technical Support.

> **Note:**
>
> After synchronizing, apply the updates to your pool as soon as possible to benefit from the latest updates.
>
> If you designate a new pool coordinator after synchronizing but before applying updates to the hosts in the pool, you must synchronize again with the new pool coordinator before you can update the pool.
>
> Do not synchronize your XenServer pool while the pool is in the process of being updated.

Understand the guidance categories and update tasks

Some tasks (such as evacuating or rebooting your hosts) might be required before and after applying updates to your pool. Sometimes, there are no update tasks required.

Guidance categories

XenServer tries to minimize the disruption to your VMs that these tasks might cause by categorizing the tasks into **Mandatory**, **Recommended**, **Full-effectiveness**, and **Live patch**. These categorizations enable you to judge whether an update task that might cause downtime or minor disruption for your hosts or VMs is necessary for your environment and risk profile.

Updates can have tasks listed in more than one of these categories. For example, an update might require that you restart the host to get the full-effectiveness of the update, but recommend restarting the toolstack to get most of the benefit of the update with less potential disruption to the pool.

During the update process, you can choose to carry out one of the following three levels of tasks:

1. Mandatory
2. Mandatory + Recommended
3. Mandatory + Recommended + Full-effectiveness

Mandatory Mandatory tasks *must* be performed after an update otherwise the system might fail at runtime. These actions are required to enable critical fixes and ensure that your environment is secure and stable. You cannot opt out of mandatory tasks.

Recommended Recommended tasks are the tasks that we recommend you perform to get the benefit of the majority of features and fixes delivered in the updates. If you choose not to perform these tasks now, they are listed in the pending update tasks for the applicable pool, host, or VM.

Why perform the recommended tasks:

- These tasks are the ones that ensure a secure, stable XenServer environment.

Why opt out of the recommended tasks:

- After reviewing the detailed information for the updates, you judge that the risk of not fully applying these updates now is acceptable.
- The recommended tasks cause unwanted disruption to your VMs now.

Full-effectiveness Full effectiveness tasks are required to get the benefit of the related update. The updates that have full-effectiveness tasks associated with them are usually relevant only to users on certain hardware or using specific features.

Review the update information to understand whether these tasks are required for your environment. If you choose not to perform these tasks now, they are listed in the pending tasks for the applicable pool, host, or VM.

Why perform the full-effectiveness tasks:

- The updates that have full-effectiveness tasks are relevant to your hardware, environment, or configuration.

Why opt out of the full-effectiveness tasks:

- The updates that have full-effectiveness guidance are not relevant to your hardware, environment, or configuration.
- The full-effectiveness tasks cause unwanted disruption to your VMs now.
- You do not need the benefits of these updates right now.

If the full-effectiveness tasks apply to your environment, but you have opted to defer them, plan to complete these tasks during a suitable maintenance window to maintain the stability of your environment.

Live patches Updates to certain components can include a live patch. Whether a live patch can be applied to your hosts depends on the version of the component that was installed when the hosts were last rebooted. If an update can be applied as a live patch to your hosts, the live patch guidance replaces the recommended guidance.

> **Example:**
>
> You have two pools. Pool A is updated to a recent level. Pool B has not been updated for some time. We release a new update that has the recommended update task "Restart host"and the live patch update task "Restart toolstack".
>
> In pool A, the live patch can be applied to these more up-to-date hosts. The recommended guidance shows "Restart toolstack". The less disruptive task from the live patch guidance overrides the recommended guidance.
>
> In pool B, the live patch cannot be applied to the hosts as they are at an older level. The recommended guidance shows "Restart host". The recommended guidance remains applicable. The live patch guidance is irrelevant in this case.

Sometimes only some of the fixes in an update are enabled when the update is applied as a live patch. Review the update details to understand whether you need all fixes in the update or just those fixes enabled by the live patch. You can then use this information to choose whether to perform the recommended tasks. For more information, see View available updates for your pool.

Update tasks

One or more of the following tasks might be required when applying an update. Any type of update task can be listed in any guidance category.

Update tasks for your host You only ever carry this task out *before* applying updates and you sometimes carry it out as part of the 'Reboot host' task:

Update task	xe CLI command to carry out task	Description
Evacuate host	`xe host-evacuate`	Migrate all VMs off the XenServer host or shut them down. While this task is in progress, the XenServer pool is operating at reduced capacity as one host is temporarily unavailable to run VMs.

The following tasks require actions on the updated host:

Update task	xe CLI command to carry out task	Description
Reboot host	`xe host-reboot`	The XenServer host must be restarted. Any VMs are migrated off the host and the host is restarted. While this task is in progress, the XenServer pool is operating at reduced capacity as one host is temporarily unavailable to run VMs.

Update task	xe CLI command to carry out task	Description
Reboot host on Xen live patch failure	`xe host-reboot`	Applying a Xen live patch failed. The XenServer host must be restarted to get the update to take effect. Any VMs are migrated off the host and the host is restarted. While this task is in progress, the XenServer pool is operating at reduced capacity as one host is temporarily unavailable to run VMs.
Reboot host on kernel live patch failure	`xe host-reboot`	Applying a dom0 kernel live patch failed. The XenServer host must be restarted to get the update to take effect. Any VMs are migrated off the host and the host is restarted. While this task is in progress, the XenServer pool is operating at reduced capacity as one host is temporarily unavailable to run VMs.
Restart toolstack	`xe-toolstack-restart`	The toolstack on the host must be restarted. Restart the toolstack on the updated host instead of the pool coordinator. When the toolstack is restarted on the pool coordinator, the connection to the pool is lost but when the toolstack is restarted on other pool members, there is no visible effect.

View the update tasks required for your host View the tasks required for your host before and after applying updates by using the following commands.

Get a list of the mandatory tasks for your host:

```
xe host-param-get param-name=pending-guidances uuid=<host UUID>
<!--NeedCopy-->
```

Get a list of the recommended tasks for your host:

```
xe host-param-get param-name=pending-guidances-recommended uuid=<host
    UUID>
<!--NeedCopy-->
```

Get a list of the full-effectiveness tasks for your host:

```
xe host-param-get param-name=pending-guidances-full uuid=<host UUID>
<!--NeedCopy-->
```

Update tasks for your VM Some updates provide new features for your VMs. These updates might require the following tasks on your VMs:

Update task	xe CLI command to carry out task	Description
Restart VM	`xe vm-reboot`	The VM must be restarted. While the VM is restarting, the VM is unavailable to the end user.
Restart device model	`xe vm-restart-device-models`	The device model for VMs on the updated host must be restarted. While the device model is restarting, you can't stop, start, or migrate the VM. The end user of the VM might see a slight pause and resume in their session. For the restart device model action to be supported on a Windows VM, the VM must have the XenServer VM Tools for Windows installed.

View the update tasks required for your VM Get a list of the mandatory tasks for your VM:

```
xe vm-param-get param-name=pending-guidances uuid=<VM UUID>
<!--NeedCopy-->
```

Get a list of the recommended tasks for your VM:

```
xe vm-param-get param-name=pending-guidances-recommended uuid=<VM UUID>
<!--NeedCopy-->
```

Get a list of the full-effectiveness tasks for your VM:

```
xe vm-param-get param-name=pending-guidances-full uuid=<VM UUID>
<!--NeedCopy-->
```

View available updates for your pool

Before installing updates, view the available updates for your pool and review the update tasks required. For more information about the different update tasks that might be required for an update, see Understand the guidance categories and update tasks.

Check for available updates for a particular host

Check if there are any available updates for a particular host:

```
xe host-param-get param-name=latest-synced-updates-applied uuid=<host
    UUID>
<!--NeedCopy-->
```

This command returns `yes` if there are updates available for a particular host and `no` if there are none available.

Check for available updates for all hosts

Check for available updates for all hosts by making a GET request on the HTTP endpoint `/updates`.

Get updates for all hosts by using the `wget` utility:

```
wget -O - --no-check-certificate https://<user name>:<password>@<
    coordinator IP address>/updates
<!--NeedCopy-->
```

Alternatively, you can use the HTTP client library:

```
HTTP GET
session_id: <XAPI session ID returned from login>
host_refs: <host XAPI reference>
<!--NeedCopy-->
```

The output returned is in JSON format and contains the following objects:

- `hosts`: Lists the available updates for individual hosts.
- `updates`: Lists the details of the available updates.
- `hash`: The `update_checksum` (used to ensure that you are always applying the latest available updates).

The `hosts` and `updates` objects also contain the `guidance` object which consists of the following keys:

- `mandatory`
- `recommended`
- `full`
- `livepatch`

These keys refer to the different guidance categories for update tasks. They list the update tasks required for your hosts and VMs. For more information about the different tasks that might be required for an update, see Understand the guidance categories and update tasks.

Apply updates to your pool

Before you start

- Ensure that all the hosts in your pool are online before carrying out the pool update.
- Ensure that there are no pending mandatory update tasks on any host or VM. Any mandatory update tasks pending from previous updates must be carried out before starting a new pool update. For more information, see View the update tasks required for your host and View the update tasks required for your VM.
- Disable High Availability (HA) if it is enabled:

  ```
  pool_uuid=$(xe pool-list --minimal)
  xe pool-ha-disable uuid=<pool_uuid>
  <!--NeedCopy-->
  ```

- Disable Workload Balancing (WLB) if it is enabled:

  ```
  pool_uuid=$(xe pool-list --minimal)
  xe pool-param-set wlb-enabled=false uuid=<pool_uuid>
  <!--NeedCopy-->
  ```

Install updates

To perform an update to your pool, you must apply updates on every host in the pool, starting with the pool coordinator first. Follow these steps, starting with the pool coordinator:

1. Disable the host:

   ```
   xe host-disable uuid=<host UUID>
   <!--NeedCopy-->
   ```

2. If one of the update tasks required for the update is 'Evacuate host'or 'Reboot host', evacuate the host:

   ```
   xe host-evacuate uuid=<host UUID>
   <!--NeedCopy-->
   ```

   If you can't migrate a VM to other hosts during the host evacuation, shut down or suspend the VM.

3. Apply updates to the host:

   ```
   xe host-apply-updates uuid=<host UUID> hash=<update_checksum>
   <!--NeedCopy-->
   ```

4. Get a list of the host update tasks required. For more information, see Update tasks for your host.

   Carry out the host's update tasks in the list in the following order:

   a) Restart toolstack (can be skipped if there is a 'Reboot host'to be carried out)
   b) Reboot host

5. For every running VM on the host, get a list of the VM update tasks required. For more information, see Update tasks for your VM.

   Carry out the VM's update tasks in the list in the following order:

   a) Restart device model (can be skipped if there is a 'Restart VM'to be carried out)
   b) Restart VM

6. Enable the host if it is still in a disabled state:

   ```
   xe host-enable uuid=<host UUID>
   <!--NeedCopy-->
   ```

7. For every VM which you migrated to another host using `host-evacuate` before the host update, get a list of the update tasks. For more information, see Update tasks for your VM.

   If 'Restart VM'is in the list of update tasks, shut down the VM and start it on the current updated host. Otherwise, migrate the VM back to the current updated host.

8. Resume or start the VMs that you shut down or suspended before you applied updates.

9. View the host update status:

```
xe host-param-get param-name=last-software-update uuid=<host UUID>
xe host-param-get param-name=latest-synced-updates-applied uuid=<
    host UUID>
xe host-param-get param-name=last-update-hash uuid=<host UUID>
<!--NeedCopy-->
```

Repeat the steps above to update every host in your pool.

After updating your hosts

After updating each host in your pool, carry out any remaining update tasks.

1. For every VM in your pool, get a list of the update tasks. For more information, see Update tasks for your VM.

   If 'Restart VM' is in the list of update tasks, carry it out.

2. Enable HA if you disabled it before applying updates:

   ```
   pool_uuid=$(xe pool-list --minimal)
   xe pool-ha-disable uuid=<pool_uuid>
   <!--NeedCopy-->
   ```

3. Enable WLB if you disabled it before applying updates:

   ```
   pool_uuid=$(xe pool-list --minimal)
   xe pool-param-set wlb-enabled=true uuid=<pool_uuid>
   <!--NeedCopy-->
   ```

4. If you chose to carry out only the mandatory update tasks required for a pool update, the update tasks that have not been carried out are appended to the list of pending update tasks required for your hosts. To view this list and carry out these tasks, see Update tasks for your host.

XenCenter current release

May 7, 2024

Use XenCenter YYYY.x.x to manage your XenServer 8 environment and to deploy, manage, and monitor virtual machines from your Windows desktop machine.

You can download the installer for the latest version of XenCenter from the XenServer download page.

> **Note:**
>
> XenCenter YYYY.x.x is not yet supported for use with Citrix Hypervisor 8.2 CU1 in production environments. To manage your Citrix Hypervisor 8.2 CU1 production environment, use XenCenter 8.2.7. For more information, see the XenCenter 8.2.7 documentation.
>
> You can install XenCenter 8.2.7 and XenCenter YYYY.x.x on the same system. Installing XenCenter YYYY.x.x does not overwrite your XenCenter 8.2.7 installation.

Using XenServer with Citrix products

March 14, 2024

XenServer provides features that enhances its interoperation with Citrix Virtual Apps and Desktops, Citrix DaaS, and Citrix Provisioning.

For more information about these products, see:

- Citrix Virtual Apps and Desktops product documentation
- Citrix DaaS product documentation
- Citrix Provisioning product documentation

Supported versions

You can find the versions of these products that XenServer 8 interoperates with on the Citrix website: Supported Hypervisors for Citrix Virtual Apps and Desktops (MCS) and Citrix Provisioning (PVS).

Licensing

To use XenServer for your Citrix Virtual Apps and Desktops or Citrix DaaS workloads, you must have a XenServer Premium Edition license. For more information, see Licensing.

If you already use XenServer or Citrix Hypervisor to host your Citrix Virtual Apps and Desktops workloads, you can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

You can get XenServer licenses from https://xenserver.com/buy.

XenServer features for Citrix Virtual Apps and Desktops

The following XenServer features are designed for use with Citrix Virtual Apps and Desktops, Citrix DaaS, and Citrix Provisioning:

- **Intellicache**: Using XenServer with IntelliCache makes hosted Citrix Virtual Desktop deployments more cost-effective by enabling you to use a combination of shared storage and local storage. It is of particular benefit when many VMs all share a common OS image. The load on the storage array is reduced and performance is enhanced. In addition, network traffic to and from shared storage is reduced as the local storage caches the primary image from shared storage.
- **Read caching**: Read caching improves a VM's disk performance by caching data within the host's free memory. It improves performance in a Citrix Virtual Desktops Machine Creation Services (MCS) environment where many VMs are cloned off a single base VM, as it drastically reduces the number of blocks read from disk.
- **PVS-Accelerator**: The XenServer PVS-Accelerator feature offers extended capabilities for customers using XenServer with Citrix Provisioning. PVS-Accelerator provides many benefits including data locality, improved end-user experience, accelerated VM boots and boot storms, simplified scale-out by adding more hypervisor hosts, and reduced TCO and simplified infrastructure requirements.
- **Smooth roaming support for Virtual Desktop Tablet Mode**: XenServer, in conjunction with Citrix Virtual Apps and Desktops, enables you to experience Windows 10 Continuum experience in a virtualized environment.
- **Graphics virtualization**: XenServer provides the virtual delivery of 3D professional graphics applications and workstations in XenServer through GPU Pass-through (for NVIDIA, AMD, and Intel GPUs) and hardware-based GPU sharing with NVIDIA vGPU™ and Intel GVT-g™.

Best practices

When configuring and managing your XenServer environment there are steps you can take to optimize how it works with Citrix products.

Installing and upgrading

- When first installing XenServer hosts, you can enable Intellicache to cache VM data locally and improve performance. For more information, see Intellicache.

- If you are upgrading from an earlier version of Citrix Hypervisor or XenServer, the method you use for this upgrade can depend on your Citrix Virtual Apps and Desktops workload. For more information, see Upgrade scenarios for Citrix Virtual Apps and Desktops.

Configuring your environment

- The XenServer host comes installed with a default TLS certificate. However, to use HTTPS to secure communication between XenServer and Citrix Virtual Apps and Desktops, install a certificate provided by a trusted certificate authority. For more information, see Install a TLS certificate on your host.

Memory usage

- When XenServer is first installed, it allocates a certain amount of memory to the control domain. In many Citrix Virtual Apps and Desktops environments, it is advisable to increase the amount of memory allocated to the control domain beyond this default.

  Increase the control domain memory in the following cases:

  - You are running many VMs on the server
  - You are using PVS-Accelerator
  - You are using read caching

  For information about changing the amount of control domain memory and monitoring the memory behavior, see Memory usage.

Upgrade scenarios for XenServer and Citrix Virtual Apps and Desktops

March 20, 2024

XenServer contains features and optimizations that make it an ideal hypervisor to use in your Citrix Virtual Apps and Desktops environment.

If you are using XenServer with Citrix Virtual Apps and Desktops, there are some considerations when performing your upgrade that are not covered in the main upgrade article: Upgrade from an existing version. Review both this article and the main upgrade article before starting your upgrade from Citrix Hypervisor 8.2 to XenServer 8.

> **Important:**
>
> If you use your Citrix Virtual Apps and Desktops license to license your Citrix Hypervisor 8.2 Cumulative Update 1 hosts, this license no longer applies to XenServer 8. You must instead get XenServer Premium Edition licenses to cover every CPU socket in your pool. For more information, see https://xenserver.com/buy.
>
> Existing customers can request to participate in our promotion and receive up to 10,000 XenServer Premium Edition socket licenses for free. Learn more.

Considerations when upgrading XenServer in a Citrix Virtual Apps and Desktops environment:

- XenServer hosts restart twice as part of an upgrade. At the beginning of the upgrade, you must boot your server into the installation media. At the end of the process, the installer restarts the server to complete the upgrade. VMs on these hosts must be either migrated or stopped during this time.
- The approach to use for upgrading XenServer depends on your XenServer environment, your Citrix Virtual Apps and Desktops environment, and the types of machines and applications being hosted by XenServer.
- You might be required to do some preparation in your Citrix Virtual Apps and Desktops environment before starting your XenServer upgrade.
- This article only covers use cases where the Citrix Virtual Apps and Desktops workload is hosted in the XenServer pool. Cases where you are also hosting parts of your Citrix Virtual Apps and Desktops infrastructure on VMs in the XenServer pool are not covered by this article. Factor these components in when doing your upgrade planning.
- Ensure that the version of Citrix Virtual Apps and Desktops you are using is supported for both the version of XenServer you are upgrading from and the version you are upgrading to. For more information, see Supported Hypervisors for Citrix Virtual Apps and Desktops (MCS) and Citrix Provisioning (PVS).
- The time it takes to do the upgrade and the potential for service outage depends on your upgrade approach. The full upgrade of an entire pool might take multiple hours to complete.
- This article assumes that the time to fully upgrade a single XenServer host is 35 minutes. This host upgrade time includes the upgrade process and any required restarts.

The approaches described in this article aim to guide you to an upgrade method that reduces the possibility of service outage and enables the upgrade process to fit within your maintenance window. However, in some cases, service outages are unavoidable. If the XenServer upgrade process cannot fit within your maintenance window, you can run your pool in mixed mode for a short time between maintenance windows. However, this is not recommended. For more information, see Mixed-mode pools.

During your planned XenServer upgrade maintenance window, follow these restrictions:

- Do not attempt to reconfigure the infrastructure of the pool that is being upgraded. For example, do not add or eject hosts from the pool.
- Do not add, start, or stop any VMs in the pool that is being upgraded.
- Do not perform catalog updates during the window.

Rolling Pool Upgrade

Rolling Pool Upgrade is a XenServer feature designed to make the upgrade process easier and to minimize downtime.

The **Rolling Pool Upgrade** wizard in XenCenter guides you through the upgrade procedure and organizes the upgrade path automatically. For pools, each of the servers in the pool is upgraded in turn, starting with the pool coordinator. Before starting an upgrade, the wizard conducts a series of prechecks. These prechecks ensure certain pool-wide features, such as high availability, are temporarily disabled and that each server in the pool is prepared for upgrade. Only one server is offline at a time. Any running VMs are automatically migrated off each server before the upgrade is installed on that server.

You can use Rolling Pool Upgrade for many of the Citrix Virtual Apps and Desktops use cases outlined in this article. For each, the upgrade time is the same: the number of hosts in the pool multiplied by the upgrade time for a single host. (**N x 35 minutes**). The potential for VM outage depends on your Citrix Virtual Apps and Desktops workload and XenServer pool setup.

Even if you intend to use Rolling Pool Upgrade to upgrade your XenServer pool, review the information for your specific environment to ensure that you understand the Citrix Virtual Apps and Desktops prerequisite actions, any special considerations, and the behavior to expect.

Use cases

This article identifies several broad use cases. For each of these use cases, we assume that the XenServer pool hosts only one type of Citrix Virtual Apps and Desktops workload. If your pool contains a mix of different types of workload, review all the cases that apply to your pool to decide what your preferred upgrade approach is.

First, consider how your XenServer environment is configured:

- XenServer pool with shared storage

  In a XenServer pool with one or more shared storage repositories (SRs), the VM disks can be hosted on this shared storage, which enables the VMs to migrate between hosts during the upgrade. This configuration can reduce or remove the need for VM downtime.

- XenServer pool without shared storage or a standalone host

In a XenServer pool without shared storage or on a standalone XenServer host, the VMs can't migrate during the upgrade process. When the host reboots as part of the upgrade, you must shut down the VMs.

XenServer pool with shared storage

If you are upgrading a pool where the VM disks are located on shared storage, you can evacuate VMs from each XenServer host in the pool while it is upgraded.

Most use cases on this type of pool can be upgraded by using Rolling Pool Upgrade. However, the prerequisite actions required in Citrix Virtual Apps and Desktops and the outage behavior is different depending on your workload.

Consider what type of Citrix Virtual Apps and Desktops workload is hosted in your pool:

- Unassigned single-session desktops
- Other workloads

XenServer pool without shared storage or a standalone host

If you are upgrading a pool where the VM disks are located on local storage or you have a single host in your pool, the VMs cannot be migrated off the XenServer hosts while they are upgraded. In these cases, the VMs must be shut down for the duration of the host or pool upgrade. Some outage to your virtual apps and desktops is unavoidable in these cases.

Consider what type of Citrix Virtual Apps and Desktops workload is hosted in your pool:

- Assigned desktops
- Other workloads

Case 1: Single-session desktops running on a pool with shared storage

This use case covers XenServer pools with shared storage whose primary workload is single-session virtual desktops with the random machine allocation type. Machines of this type must be managed by either Citrix Provisioning or by Machine Creation Services.

For any workload that is managed by Citrix Virtual Apps and Desktops, including those that are power managed by Citrix Provisioning and Machine Creation Services, you cannot maintain a complete workload while the upgrade is being performed. Power management of machines can be problematic during the upgrade process and you cannot disable power management without also disabling new session creation.

Recommended options for upgrade:

- Rolling Pool Upgrade
  - Estimated upgrade time: The number of hosts in the pool multiplied by the upgrade time for a single host. (**N x 35 minutes**)
  - Outage behavior: All machines are in Citrix Virtual Apps and Desktops maintenance mode for the entire upgrade time.

If possible, make the workload available from other XenServer pools with capacity during the upgrade of this pool. This approach might cause reduced capacity during the upgrade. If you do not have capacity for the workload on your other XenServer hosts and pools, we recommend that you declare an outage for all machines in your workload.

Rolling Pool Upgrade (1)

Review the steps and guidance in Before you start.

1. Put all the machines in the pool in maintenance mode. If all of the machines are using the same connection, you can put the entire machine catalog in maintenance mode.
2. Notify all impacted users of the impending outage.
   - If sessions are still running on the machines in this pool, ask users to log off or force their sessions to end.
   - Inform users that after they log off they can't log in again until full service is resumed.
3. In XenCenter, start the Rolling Pool Upgrade wizard and choose automatic mode. For more information, see Rolling Pool Upgrade by using XenCenter.

   When the upgrade is complete, any VMs that were suspended as part of the Rolling Pool Upgrade are restarted.
4. Take the machines out of maintenance mode.

   New sessions can now be started and full service resumed.

Case 2: Other workloads running on a pool with shared storage

This use case covers XenServer pools with shared storage whose primary workload is either single-session virtual desktops with the assigned machine allocation type or multiple-session virtual applications with the random machine allocation type.

Recommended options for upgrade:

- Rolling Pool Upgrade

    - Estimated upgrade time: The number of hosts in the pool multiplied by the upgrade time for a single host. (**N x 35 minutes**)
    - Outage behavior: No service outage

Rolling Pool Upgrade (2)

Review the steps and guidance in Before you start.

1. Ensure that the pool has enough capacity to run your workload with one fewer host in the pool. During the upgrade process, each host is removed one at a time. The remaining hosts must be able to run all required VMs.

   If there is not enough capacity in the pool, some machines might not be available during the upgrade process. If possible, you can suspend any non-critical VMs during the upgrade process.

2. Ensure that all machines provided by the XenServer pool are powered on and are registered with Citrix Virtual Apps and Desktops in the relevant Delivery Groups.

   - For unmanaged machines:

     - Use XenCenter to confirm that all VMs are powered on.
     - Do not perform any manual power actions during the upgrade process.

   - For power-managed machines:

     - Ensure that all machines are powered on (by using XenCenter, Citrix Studio, or Web Studio).
     - **To enable new sessions to start during the upgrade process:**
       * Do not set the machines in maintenance mode.
       * Do not perform any manual power actions during the upgrade process.
       * Disable any power management schemes that might suspend machines.
       * Ensure that there are no other processes that might power off or suspend the machines.
     - **If it's acceptable for new sessions to be unable to start during the upgrade:**
       * Put the hosting connection in maintenance mode. For more information, see Turn maintenance mode on or off for a connection.
       * Inform end users that if they log off they cannot reconnect for the time of the upgrade.

     For more information, see Power managed machines in a delivery group.

   - For machines managed by Machine Creation Services

     - Follow the same guidance as for power-managed machines in the preceding list item.
     - In addition, do not attempt to create new machines during the entire upgrade period.

3. In XenCenter, start the Rolling Pool Upgrade wizard and choose automatic mode. For more information, see Rolling Pool Upgrade by using XenCenter.
4. Restore your environment operations to their usual configuration.
   - Remove any maintenance mode flags set in earlier steps.
   - Revert any power management scheme adjustments made in the earlier steps.

Case 3: Assigned desktops running on a pool with local storage or on a standalone host

This use case covers XenServer standalone hosts or pools that do not have shared storage whose primary workload is either single-session virtual desktops with the assigned machine allocation type.

Recommended options for upgrade:

- Rolling Pool Upgrade Use RPU in automatic mode in a single maintenance window. This requires all users to have outage for the entire upgrade, but has a lower administration overhead for a pool
  - Estimated upgrade time: The number of hosts in the pool multiplied by the upgrade time for a single host. (**N x 35 minutes**)
  - Outage behavior: All machines are in Citrix Virtual Apps and Desktops maintenance mode for the entire upgrade time.
- Manual upgrade This mode provides the least outage for each user during the upgrade, but is more involved for the administrator
  - Estimated upgrade time: Two times the upgrade time for a single host. (**Approximately 70 minutes**)
  - Outage behavior: Each desktop is unavailable during the upgrade time for their individual host. This time is typically 35 minutes.

Rolling Pool Upgrade (3)

Review the steps and guidance in Before you start.

1. Put all delivery groups or catalogs that are providing machines from the pool into maintenance mode.

   While the machines are in maintenance mode, new sessions cannot be started on machines in the pool. Existing sessions are maintained until the machines are shut down or suspended.

   For more information, see Prevent users from connecting to a machine in a delivery group.
2. Notify all affected users of the impending outage. Provide a time by which they must end their sessions and indicate when service will be restored.

3. Check for remaining sessions on impacted machines and take appropriate actions for these sessions.
4. In XenCenter, start the Rolling Pool Upgrade wizard and choose automatic mode. For more information, see Rolling Pool Upgrade by using XenCenter.

   When the upgrade is complete, any VMs that were suspended as part of the Rolling Pool Upgrade are restarted.
5. Take the machines out of maintenance mode.

   New sessions can now be started and full service resumed.

Manual upgrade (3)

You can use this manual process to upgrade the pool coordinator first then all other hosts in parallel to reduce overall outage time significantly.

> **Note:**
>
> With the parallel upgrade approach, the risk profile changes. If there is an issue during the upgrade, it might not be detected until all hosts have been upgraded and are experiencing the issue. Whereas if you upgrade your hosts sequentially, you can verify that the upgrade is successful on each host before going on to the next.

Review the steps and guidance in Before you start.

1. Ensure that all machines provided by the XenServer pool or host are turned on and are registered with Citrix Virtual Apps and Desktops in the relevant delivery groups.

   - For unmanaged machines:
     - Use XenCenter to confirm that all VMs are powered on.
     - Do not perform any manual power actions during the upgrade process.
   - For power-managed machines:
     - Ensure that all machines are powered on (using XenCenter or Studio).
     - **To enable new sessions to start during the upgrade process:**
       - Do not set the machines in maintenance mode.
       - Do not perform any manual power actions during the upgrade process.
       - Disable any power management schemes that might suspend machines.
       - Ensure that there are no other processes that might power off or suspend the machines.
     - **If it's acceptable for new sessions to be unable to start during the upgrade:**

        * Put the hosting connection in maintenance mode. For more information, see Turn maintenance mode on or off for a connection.
        * Inform end users that if they log off they cannot reconnect for the time of the upgrade.

    For more information, see Power managed machines in a delivery group.

  - For machines managed by Machine Creation Services

    - Follow the same guidance as for power-managed machines in the preceding list item.
    - In addition, do not attempt to create machines during the entire upgrade period.

2. Identify the pool coordinator and associated VMs.

3. Put the machines in the catalog on the pool coordinator host into maintenance mode.

4. Use Director, Citrix Studio, or Web Studio to send messages to users that are still connected to active sessions, warning them that their desktop is going offline for a period. This period is the upgrade time for this individual host (approximately 35 minutes).

5. Update the pool coordinator by using the xe CLI:

    a) Disable the pool coordinator. This prevents any new VMs from starting on or being migrated to the specified host.

    ```
    xe host-disable host=<uuid_or_name_label>
    ```

    b) Ensure that no VMs are running on the pool coordinator. Shut down, suspend, or migrate VMs to other hosts in the pool.

    - To shut down a VM, use the following command:

      ```
      xe vm-shutdown
      ```

    - To suspend a VM, use the following command:

      ```
      xe vm-suspend
      ```

    - To migrate a specific VM, use the following command:

      ```
      xe vm-migrate
      ```

      Migrating specified VMs to specified hosts gives you full control over the distribution of migrated VMs to other hosts in the pool.

    - To evacuate the host, use the following command:

      ```
      xe host-evacuate
      ```

      Evacuating all VMs from a host leaves the distribution of migrated VMs to XenServer.

c) Shut down the pool coordinator.

```
xe host-shutdown
```

> **Important:**
>
> You are unable to contact the pool coordinator until the upgrade of the pool coordinator is complete. Shutting down the pool coordinator causes the other hosts in the pool to enter *emergency mode*. Hosts can enter emergency mode when they in a pool whose pool coordinator has disappeared from the network and cannot be contacted after several attempts. VMs continue to run on hosts in emergency mode, but control operations are not available.

d) Boot the pool coordinator using the XenServer installation media and method of your choice (such as, USB or network).

e) Follow the XenServer installation procedure until the installer offers you the option to upgrade. Choose to upgrade.

When your pool coordinator restarts, the other hosts in the pool leave emergency mode and normal service is restored after a few minutes.

f) Start or resume any shutdown or suspended VMs.

g) Migrate any VMs that you want back to the pool coordinator.

If anything interrupts the upgrade of the pool coordinator or if the upgrade fails for any reason, do not attempt to proceed with the upgrade. Reboot the pool coordinator and restore to a working version.

6. After the pool coordinator is upgraded, take the machines on the pool coordinator out of maintenance mode in Citrix Studio or Web Studio.
7. Complete the following steps in parallel for all remaining hosts in the pool:

a) Put the machines in the catalog on the host into maintenance mode.

b) Use Director, Citrix Studio, or Web Studio to send messages to users that are still connected to active sessions, warning them that their desktop is going offline for a period. This period is the upgrade time for this individual host (approximately 35 minutes).

c) Disable the host by using the xe CLI.

```
xe host-disable host-selector=<host_selector_value>
```

d) Ensure that no VMs are running on the host. Shut down, suspend, or migrate VMs to other hosts in the pool.

- To shut down a VM, use the following command:

```
xe vm-shutdown
```

- To suspend a VM, use the following command:

```
xe vm-suspend
```

- To migrate a specific VM, use the following command:

```
xe vm-migrate
```

Migrating specified VMs to specified hosts gives you full control over the distribution of migrated VMs to other hosts in the pool.

- To evacuate the host, use the following command:

```
xe host-evacuate
```

Evacuating all VMs from a host leaves the distribution of migrated VMs to XenServer.

e) Shut down the host.

```
xe host-shutdown
```

f) Boot the host using the XenServer installation media and method of your choice (such as, USB or network).

g) Follow the XenServer installation procedure until the installer offers you the option to upgrade. Choose to upgrade.

h) After the host upgrade is complete, start or resume any shutdown or suspended VMs.

i) Migrate any VMs that you want back to the host.

If the upgrade of a subordinate host fails or is interrupted, you do not have to revert. Run the command `xe host-forget` in the pool to forget that host. Reinstall XenServer on the host, and then join it, as a new host, to the pool using the command `xe pool-join`.

8. After the XenServer hosts are updated, take the machines out of maintenance mode in Citrix Studio or Web Studio.

Case 4: Other workloads running on a pool with local storage or on a standalone host

This use case covers XenServer pools with shared storage whose primary workload is single-session virtual desktops or multiple-session virtual applications with the random machine allocation type.

For any workload that is managed by Citrix Virtual Apps and Desktops, including those that are power managed by Citrix Provisioning and Machine Creation Services, you cannot maintain a complete workload while the upgrade is being performed. Power management of machines can be problematic during the upgrade process and you cannot disable power management without also disabling new session creation.

Recommended options for upgrade:

- Rolling Pool Upgrade
  - Estimated upgrade time: The number of hosts in the pool multiplied by the upgrade time for a single host. (**N x 35 minutes**)
  - Outage behavior: All machines are in Citrix Virtual Apps and Desktops maintenance mode for the entire upgrade time.
- Manual upgrade
  - Estimated upgrade time: Two times the upgrade time for a single host. (**Approximately 70 minutes**)
  - Outage behavior: All machines are in Citrix Virtual Apps and Desktops maintenance mode for the entire upgrade time.

If possible, make the workload available from other XenServer pools with capacity during the upgrade of this pool. This approach might cause reduced capacity during the upgrade. If you do not have capacity for the workload on your other XenServer hosts and pools, we recommend that you declare an outage for all machines in your workload.

Rolling Pool Upgrade (4)

Review the steps and guidance in Before you start.

1. Put all the machines in the pool in maintenance mode. If all of the machines are using the same connection, you can put the entire machine catalog in maintenance mode.
2. Notify all impacted users of the impending outage.
   - If sessions are still running on the machines in this pool, ask users to log off or force their sessions to end.
   - Inform users that after they log off they can't log in again until full service is resumed.
3. In XenCenter, start the Rolling Pool Upgrade wizard and choose automatic mode. For more information, see Rolling Pool Upgrade by using XenCenter.

   When the upgrade is complete, any VMs that were suspended as part of the Rolling Pool Upgrade are restarted.
4. Take the machines out of maintenance mode.

   New sessions can now be started and full service resumed.

Manual upgrade (4)

You can use this manual process to upgrade the pool coordinator first then all other hosts in parallel to reduce overall outage time significantly.

> **Note:**
>
> With the parallel upgrade approach, the risk profile changes. If there is an issue during the upgrade, it might not be detected until all hosts have been upgraded and are experiencing the issue. Whereas if you upgrade your hosts sequentially, you can verify that the upgrade is successful on each host before going on to the next.

Review the steps and guidance in Before you start.

1. Ensure that all machines provided by the XenServer pool or host are turned on and are registered with Citrix Virtual Apps and Desktops in the relevant delivery groups.

    - For unmanaged machines:

        - Use XenCenter to confirm that all VMs are powered on.
        - Do not perform any manual power actions during the upgrade process.

    - For power-managed machines:

        - Ensure that all machines are powered on (using XenCenter or Studio).
        - **To enable new sessions to start during the upgrade process:**
            * Do not set the machines in maintenance mode.
            * Do not perform any manual power actions during the upgrade process.
            * Disable any power management schemes that might suspend machines.
            * Ensure that there are no other processes that might power off or suspend the machines.
        - **If it's acceptable for new sessions to be unable to start during the upgrade:**
            * Put the hosting connection in maintenance mode. For more information, see Turn maintenance mode on or off for a connection.
            * Inform end users that if they log off they cannot reconnect for the time of the upgrade.

        For more information, see Power managed machines in a delivery group.

    - For machines managed by Machine Creation Services

        - Follow the same guidance as for power-managed machines in the preceding list item.
        - In addition, do not attempt to create machines during the entire upgrade period.

2. Identify the pool coordinator and associated VMs.

3. Put the machines in the catalog on the pool coordinator host into maintenance mode.

4. Use Director, Citrix Studio, or Web Studio to send messages to users that are still connected to active sessions, warning them that their desktop is going offline for a period. This period is the upgrade time for this individual host (approximately 35 minutes).
5. Update the pool coordinator by using the xe CLI:

   a) Disable the pool coordinator. This prevents any new VMs from starting on or being migrated to the specified host.

   ```
   xe host-disable host=<uuid_or_name_label>
   ```

   b) Ensure that no VMs are running on the pool coordinator. Shut down, suspend, or migrate VMs to other hosts in the pool.

   - To shut down a VM, use the following command:

     ```
     xe vm-shutdown
     ```

   - To suspend a VM, use the following command:

     ```
     xe vm-suspend
     ```

   - To migrate a specific VM, use the following command:

     ```
     xe vm-migrate
     ```

     Migrating specified VMs to specified hosts gives you full control over the distribution of migrated VMs to other hosts in the pool.

   - To evacuate the host, use the following command:

     ```
     xe host-evacuate
     ```

     Evacuating all VMs from a host leaves the distribution of migrated VMs to XenServer.

   c) Shut down the pool coordinator.

   ```
   xe host-shutdown
   ```

   > **Important:**
   >
   > You are unable to contact the pool coordinator until the upgrade of the pool coordinator is complete. Shutting down the pool coordinator causes the other hosts in the pool to enter *emergency mode*. Hosts can enter emergency mode when they in a pool whose pool coordinator has disappeared from the network and cannot be contacted after several attempts. VMs continue to run on hosts in emergency mode, but control operations are not available.

   d) Boot the pool coordinator using the XenServer installation media and method of your choice (such as, USB or network).

e) Follow the XenServer installation procedure until the installer offers you the option to upgrade. Choose to upgrade.

   When your pool coordinator restarts, the other hosts in the pool leave emergency mode and normal service is restored after a few minutes.

f) Start or resume any shutdown or suspended VMs.

g) Migrate any VMs that you want back to the pool coordinator.

If anything interrupts the upgrade of the pool coordinator or if the upgrade fails for any reason, do not attempt to proceed with the upgrade. Reboot the pool coordinator and restore to a working version.

6. After the pool coordinator is upgraded, take the machines on the pool coordinator out of maintenance mode in Citrix Studio or Web Studio.

7. Complete the following steps in parallel for all remaining hosts in the pool:

   a) Put the machines in the catalog on the host into maintenance mode.

   b) Use Director, Citrix Studio, or Web Studio to send messages to users that are still connected to active sessions, warning them that their desktop is going offline for a period. This period is the upgrade time for this individual host (approximately 35 minutes).

   c) Disable the host by using the xe CLI.

   ```
   xe host-disable host-selector=<host_selector_value>
   ```

   d) Ensure that no VMs are running on the host. Shut down, suspend, or migrate VMs to other hosts in the pool.

   - To shut down a VM, use the following command:

     ```
     xe vm-shutdown
     ```

   - To suspend a VM, use the following command:

     ```
     xe vm-suspend
     ```

   - To migrate a specific VM, use the following command:

     ```
     xe vm-migrate
     ```

     Migrating specified VMs to specified hosts gives you full control over the distribution of migrated VMs to other hosts in the pool.

   - To evacuate the host, use the following command:

     ```
     xe host-evacuate
     ```

Evacuating all VMs from a host leaves the distribution of migrated VMs to XenServer.

e) Shut down the host.

```
xe host-shutdown
```

f) Boot the host using the XenServer installation media and method of your choice (such as, USB or network).

g) Follow the XenServer installation procedure until the installer offers you the option to upgrade. Choose to upgrade.

h) After the host upgrade is complete, start or resume any shutdown or suspended VMs.

i) Migrate any VMs that you want back to the host.

If the upgrade of a subordinate host fails or is interrupted, you do not have to revert. Run the command `xe host-forget` in the pool to forget that host. Reinstall XenServer on the host, and then join it, as a new host, to the pool using the command `xe pool-join`.

8. After the XenServer hosts are updated, take the machines out of maintenance mode in Citrix Studio or Web Studio.

Mixed-mode pools

A mixed-mode pool is one where hosts in the pool are using different versions of XenServer. Do not operate your pool in mixed-mode (with multiple versions of XenServer) for longer than necessary, as the pool operates in a degraded state during upgrade. In this degraded state, certain VM, SR, VDI, and host operations are blocked. VMs that have run on a host at the higher version of XenServer cannot be migrated to or started on a host at the lower version of XenServer.

Mixed-mode pools are not supported for standard usage and are only supported as a transitional state during the upgrade of a pool. If you experience an issue while running in mixed mode, Technical Support will ask you to complete your pool upgrade and then reproduce the issue in a non-mixed pool.

After reviewing the upgrade options for your Citrix Virtual Apps and Desktops environment, your planned XenServer upgrade path might take longer than the available maintenance window. If possible, extend the maintenance window to enable your XenServer upgrade to complete within it. If this is not possible, you can choose to run the pool in mixed mode until your next maintenance window. However, running your pool in mixed mode increases the likelihood of unexpected behaviors or issues that might cause you to need an emergency maintenance window instead. Plan to minimize the time your pool spends in mixed mode.

If your Citrix Virtual Apps and Desktops environment is running temporarily on top of a mixed-mode XenServer pool, be aware of the following behaviors:

- For Pooled Desktop workloads that require the VMs to restart before they are reused, the VMs are restarted only on the hosts that are running the newer version of XenServer. The effective capacity of the pool is restricted. Depending on how many of the hosts in your pool have been upgraded, there might be insufficient capacity for all required VMs to be restarted. This behavior can result in failures and some Citrix Virtual Apps and Desktops users might not be able to access their required sessions.
- If you have dedicated machines using local storage that are located on hosts running the older version of XenServer, these VMs can be stopped, but they cannot be restarted until the upgrade is complete and the pool is no longer in mixed mode.

IntelliCache

February 7, 2024

> **Note:**
>
> This feature is only supported when using XenServer Premium Edition with Citrix Virtual Desktops.
>
> Intellicache is not supported for VMs using a GFS2 or XFS SR.

Using XenServer with *IntelliCache* makes hosted Virtual Desktop Infrastructure deployments more cost-effective by enabling you to use a combination of shared storage and local storage. It is of particular benefit when many Virtual Machines (VMs) all share a common OS image. The load on the storage array is reduced and performance is enhanced. In addition, network traffic to and from shared storage is reduced as the local storage caches the primary image from shared storage.

IntelliCache works by caching data from a VMs parent VDI in local storage on the VM host. This local cache is then populated as data is read from the parent VDI. When many VMs share a common parent VDI, a VM can use the data read into the cache from another VM. Further access to the primary image on shared storage is not required.

A thin-provisioned, local SR is required for IntelliCache. Thin provisioning is a way of optimizing the use of available storage. This approach allows you to make more use of local storage instead of shared storage. It relies on on-demand allocation of blocks of data. In other approaches, all blocks are allocated up front.

> **Important:**
>
> Thin Provisioning changes the default local storage type of the host from LVM to EXT4. Thin Provisioning **must be** enabled in order for Citrix Virtual Desktops local caching to work properly.

Thin Provisioning allows the administrator to present more storage space to the VMs connecting to the Storage Repository (SR) than is available on the SR. There are no space guarantees, and allocation of a LUN does not claim any data blocks until the VM writes data.

> **Warning:**
>
> Thin-provisioned SRs may run out of physical space, as the VMs within can grow to consume disk capacity on demand. IntelliCache VMs handle this condition by automatically falling back to shared storage when the local SR cache is full. Do not mix traditional virtual machines and IntelliCache VMs on the same SR, as IntelliCache VMs can grow quickly in size.

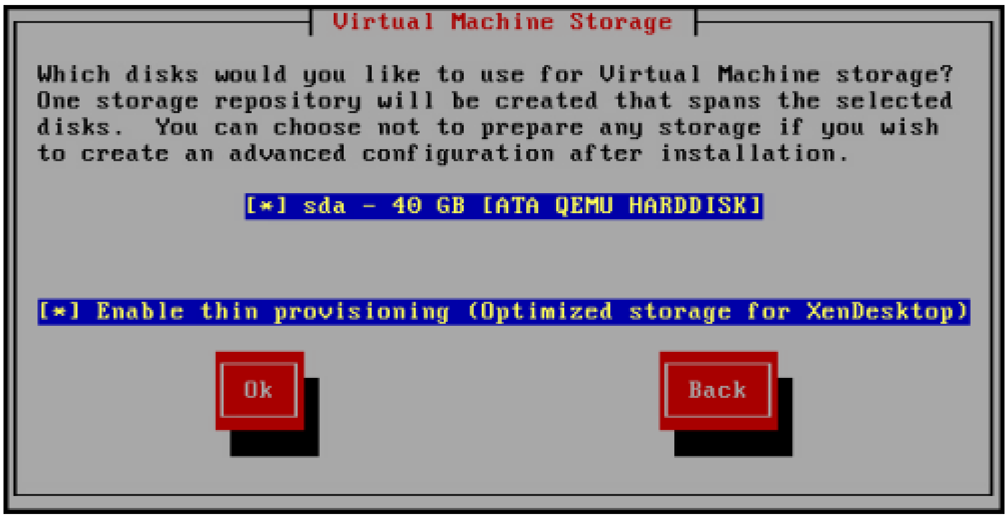
IntelliCache deployment

IntelliCache must be enabled either during host installation or be enabled manually on a running host using the CLI.

We recommend that you use a high performance local storage device to ensure the fastest possible data transfer. For example, use a Solid State Disk or a high performance RAID array. Consider both data throughput and storage capacity when sizing local disks. The shared storage type, used to host the source Virtual Disk Image (VDI), must be NFS or EXT3/EXT4 based.

Enable on host installation

To enable IntelliCache during host installation, on the **Virtual Machine Storage** screen, select **Enable thin provisioning**. This option selects the host's local SR to be the one to be used for the local caching of VM VDIs.



```
Virtual Machine Storage

Which disks would you like to use for Virtual Machine storage?
One storage repository will be created that spans the selected
disks.  You can choose not to prepare any storage if you wish
to create an advanced configuration after installation.

            [*] sda - 40 GB [ATA QEMU HARDDISK]

[*] Enable thin provisioning (Optimized storage for XenDesktop)

        Ok                              Back
```

Convert an existing host to use thin provisioning

To delete an existing LVM local SR, and replace it with a thin-provisioned EXT3/EXT4 SR, enter the following commands.

> **Warning:**
>
> These commands remove your existing local SR, and VMs on the SR are permanently deleted.

```
    localsr=`xe sr-list type=lvm host=hostname params=uuid --minimal`
        echo localsr=$localsr
        pbd=`xe pbd-list sr-uuid=$localsr params=uuid --minimal`
        echo pbd=$pbd
        xe pbd-unplug uuid=$pbd
        xe pbd-destroy uuid=$pbd
        xe sr-forget uuid=$localsr
        sed -i "s/'lvm'/'ext'/" /etc/firstboot.d/data/default-storage.
           conf
        rm -f /var/lib/misc/ran-storage-init
        systemctl restart storage-init.service
        xe sr-list type=ext
<!--NeedCopy-->
```

To enable local caching, enter the following commands:

```
    xe host-disable host=hostname
        localsr=`xe sr-list type=ext host=hostname params=uuid --
           minimal`
        xe host-enable-local-storage-caching host=hostname sr-uuid=
           $localsr
        xe host-enable host=hostname
<!--NeedCopy-->
```

VM behavior with Intellicache

The VDI flag `on-boot` dictates the behavior of a VM VDI when the VM is booted and the VDI flag `allow-caching` dictates the caching behavior.

The values to use for these parameters depends on the type of VM you are creating and what its intended use is:

- **For shared or randomly allocated machines:**
  - Set the `on-boot` parameter to `reset`.
  - Set the `allow-caching` parameter to **`true`**

  For example:

```
xe vdi-param-set uuid=vdi_uuid on-boot=reset allow-caching=true
<!--NeedCopy-->
```

On VM boot, the VDI is reverted to the state it was in at the previous boot. All changes while the VM is running are lost when the VM is next booted. New VM data is written only to local storage. There are no writes to shared storage. This approach means that the load on shared storage is reduced. However the VM cannot be migrated between hosts.

Select this option if you plan to deliver standardized desktops to which users cannot make permanent changes.

- **For static or dedicated machines:**
  - Set the `on-boot` parameter to `persist`.
  - Set the `allow-caching` parameter to `true`

  For example:

```
xe vdi-param-set uuid=vdi_uuid on-boot=persist allow-caching=true
<!--NeedCopy-->
```

On VM boot, the VDI is in the state it was left in at the last shutdown. New VM data is written to both local and shared storage. Reads of cached data do not require I/O traffic to shared storage so the load on shared storage is reduced. VM migration to another host is permitted and the local cache on the new host is populated as data is read.

Select this option if you plan to allow users to make permanent changes to their desktops.

> **Note:**
>
> For VMs whose VDIs are located on a GFS2 SR, the VM on-boot behavior is different to VMs with VDIs on other types of SRs. For VDIs on a GFS2 SR, the on-boot option is applied on VM shutdown, not on VM boot.

Implementation details and troubleshooting

Q: Is IntelliCache compatible with live migration and High Availability?

A: You can use live migration and High Availability with IntelliCache when virtual desktops are in Private mode, that is when `on-boot=persist`

> **Warning:**
>
> A VM cannot be migrated if any of its VDIs have caching behavior flags set to `on-boot=reset` and `allow-caching=true`. Migration attempts for VMs with these properties fail.

Q: Where does the local cache live on the local disk?

A: The cache lives in a Storage Repository (SR). Each host has a configuration parameter (called local-cache-sr) indicating which (local) SR is to be used for the cache files. Typically, this SR is an EXT3/EXT4 type SR. When you run VMs with IntelliCache, you see files inside the SR with names `uuid.vhdcache`. This file is the cache file for the VDI with the given UUID. These files are not displayed in XenCenter –the only way of seeing them is by logging into dom0 and listing the contents of `/var/run/sr-mount/sr-uuid`

Q: How do I specify a particular SR for use as the cache?

A: The host object field `local-cache-sr` references a local SR. You can view its value by running the following command:

```
xe sr-list params=local-cache-sr,uuid,name-label
<!--NeedCopy-->
```

This field is set either:

- After host installation, if you have chosen "Enable thin provisioning" option in the host installer, or
- By running `xe host-enable-local-storage-caching host=host sr-uuid=sr`. The command requires the specified host to be disabled. Shut down the VMs when you use this command.

The first option uses the EXT3/EXT4 type local SR and is created during host installation. The second option uses the SR that is specified on the command-line.

> **Warning:**
>
> These steps are only necessary for users who have configured more than one local SR.

Q: When is the local cache deleted?

A: A VDI cache file is only deleted when the VDI itself is deleted. The cache is reset when a VDI is attached to a VM (for example on VM start). If the host is offline when you delete the VDI, the SR synchronization that runs on startup garbage collects the cache file.

> **Note:**
>
> The cache file is not deleted from the host when a VM migrates to a different host or is shut down.

PVS-Accelerator

April 24, 2024

The XenServer PVS-Accelerator feature offers extended capabilities for customers using XenServer with Citrix Provisioning. Citrix Provisioning is a popular choice for image management and hosting for Citrix Virtual Apps and Desktops or Citrix DaaS. PVS-Accelerator dramatically improves the already excellent combination of XenServer and Citrix Provisioning. Some of the benefits that this new feature provides include:

- **Data locality:** Use the performance and locality of memory, SSD, and NVM devices for read requests, while substantially reducing network utilization.
- **Improved end-user experience:** Data locality enables a reduction in the read I/O latency for cached target devices (VMs), further accelerating end-user applications.
- **Accelerated VM boots and boot storms:** Reduced read I/O-latency and improved efficiency can accelerate VM boot times and enable faster performance when many devices boot up within a narrow time frame.
- **Simplified scale-out by adding more hypervisor hosts:** Fewer Citrix Provisioning servers may be needed as the storage load is efficiently dispersed across all XenServer hosts. Peak loads are handled using the cache within originating hosts.
- **Reduced TCO and simplified infrastructure requirements:** Fewer Citrix Provisioning servers means a reduction in hardware and license requirements, in addition to reduced management overhead. Freed up capacity is available for workloads.

> **Notes:**
>
> PVS-Accelerator is available for XenServer Premium Edition customers. To use the PVS-Accelerator feature, upgrade the Citrix License Server to version 11.14 or later.
>
> To use PVS-Accelerator with UEFI-enabled VMs, ensure that you are using Citrix Provisioning 1906 or later.

This is an embedded video. Click the link to watch the video

How does PVS-Accelerator work

PVS-Accelerator employs a Proxy mechanism that resides in the Control Domain (dom0) of XenServer. When this feature is enabled, Citrix Provisioning targets device (VM) read requests are cached directly on the XenServer host machine. These requests are cached in physical memory or a storage repository. When subsequent VMs on that XenServer host make the same read request, the virtual disk is streamed directly from cache, not from the Citrix Provisioning server. Removing the need to stream from the Citrix Provisioning server reduces network utilization and processing on the server considerably. This approach results in a substantial improvement in VM performance.

Considerations

Consider the following when using the PVS-Accelerator feature:

- Citrix Provisioning target devices are aware of their proxy status. No additional configuration is required once the capability is installed.
- In environments where multiple Citrix Provisioning servers are deployed with the same VHD, but have different file system timestamps, data might be cached multiple times. Due to this limitation, we recommend using VHDX format, rather than VHD for virtual disks.
- Do not use a large port range for PVS server communication. Setting a range of more than 20 ports on the PVS server is rarely necessary. A large port range can slow packet processing and increase the boot time of the XenServer control domain when using PVS-Accelerator.
- After you start a VM with PVS-Accelerator enabled, the caching status for the VM is displayed in XenCenter:
  - In the **PVS** tab of the pool or the host
  - In the **General** tab for the VM
- You cannot run more than 200 PVS-Accelerator-enabled VMs on a XenServer host.
- Customers can confirm the correct operation of the PVS-Accelerator using RRD metrics on the host's **Performance** tab in XenCenter. For more information, see Monitor and manage your deployment.
- PVS-Accelerator requires Citrix Provisioning 7.13 or later.
- To use PVS-Accelerator with UEFI-enabled VMs, ensure that you are using Citrix Provisioning 1906 or later.
- PVS-Accelerator is available for XenServer Premium Edition customers.
- PVS-Accelerator requires License Server 11.14 or later.
- PVS-Accelerator uses capabilities of OVS and is therefore not available on hosts that use Linux Bridge as the network back-end.
- PVS-Accelerator works on the first virtual network interface (VIF) of a cached VM. Therefore, connect the first VIF to the Citrix Provisioning storage network for caching to work.
- PVS-Accelerator can currently not be used on network ports which enforce that IPs are bound to certain MAC addresses. This switch functionality might be called "IP Source Guard" or similar. In such environments, PVS targets fail to boot with error 'Login request time out!' after enabling PVS-Accelerator.

Enable PVS-Accelerator

Customers must complete the following configuration settings in XenServer and in Citrix Provisioning to enable the PVS-Accelerator feature:

1. Configure PVS-Accelerator in XenServer by using XenCenter or the xe CLI. This configuration includes adding a Citrix Provisioning site and specifying the location for Citrix Provisioning cache storage.
    - For CLI instructions, see *Configuring PVS-Accelerator in XenServer by using the CLI* in the following section.
    - For information about configuring PVS-Accelerator using XenCenter, see PVS-Accelerator in the XenCenter documentation.
2. After configuring PVS-Accelerator in XenServer, complete the cache configuration for the PVS Site using the PVS UI. For detailed instructions, see Completing the cache configuration in Citrix Provisioning.

Configuring ports

Citrix Provisioning Services uses the following ports:

- 6901, 6902, 6905: Used for provisioning server outbound communication (packets destined for the target device)
- 6910: Used for target device logon with Citrix Provisioning Services
- Configurable target device port. The default port is 6901.
- Configurable server port range. The default range is 6910-6930.

For information about the ports used by Citrix Provisioning Services, see Communication ports used by XenServer.

The configured port range in XenServer must include all the ports in use. For example, use 6901-6930 for the default configuration.

> **Note:**
>
> Do not use a large port range for PVS server communication. Setting a range of more than 20 ports on the PVS server is rarely necessary. A large port range can slow packet processing and increase the boot time of the XenServer control domain when using PVS-Accelerator.

Configure PVS-Accelerator in XenServer by using the CLI

1. Run the following command to create a Citrix Provisioning site configuration on XenServer:

```
PVS_SITE_UUID=$(xe pvs-site-introduce name-label=My PVS Site)
```

2. For each host in the pool, specify what cache to use. You can choose to store the cache on a storage repository (SR) or in the Control Domain Memory.

Configure cache storage on a storage repository Consider the following characteristics when choosing a storage repository (SR) for cache storage:

Advantages:

- Most recently read data is cached in the memory on a best effort basis. Accessing the data can be as fast as using the Control Domain memory.
- The cache can be much larger when it is on an SR. The cost of the SR space is typically a fraction of the cost of the memory space. Caching on an SR can take more load off the Citrix Provisioning server.
- You don't have to modify the Control Domain memory setting. The cache automatically uses the memory available in the Control Domain and never causes the Control Domain to run out of memory.
- The cache VDIs can be stored on shared storage. However, this choice of storage rarely makes sense. This approach only makes sense where the shared storage is significantly faster than the Citrix Provisioning server.
- You can use either a file-based or a block-based SR for cache storage.

Disadvantages:

- If the SR is slow and the requested data isn't in the memory tier, the caching process can be slower than a remote Citrix Provisioning server.
- Cached VDIs that are stored on shared storage cannot be shared between hosts. A cached VDI is specific to one host.

Perform the following steps to configure cache storage on a Storage Repository:

1. Run the following command to find the UUID of the SR that to use for caching:

```
xe sr-list name-label=Local storage host=host-name-label --minimal
   )
<!--NeedCopy-->
```

2. Create the cache-storage.

```
xe pvs-cache-storage-create host=host-name-label pvs-site-uuid=
   PVS_SITE_UUID \
        sr-uuid=SR_UUID size=10GiB
<!--NeedCopy-->
```

> **Note:**
>
> When selecting a Storage Repository (SR), the feature uses up to the specified cache size on the SR. It also implicitly uses available Control Domain memory as a best effort cache tier.

Configuring cache storage in the control domain memory Consider the following characteristics when choosing the Control Domain memory for cache storage:

Advantages:

Using memory means consistently fast Read/Write performance when accessing or populating the cache.

Disadvantages:

- Hardware must be sized appropriately as the RAM used for cache storage is not available for VMs.
- Control Domain memory must be extended **before** configuring cache storage.

  > **Note:**
  >
  > If you choose to store the cache in the Control Domain memory, the feature uses up to the specified cache size in Control Domain memory. This option is only available after extra memory has been assigned to the Control Domain. For information about increasing the Control Domain memory, see Change the amount of memory allocated to the control domain.

  After you increase the amount of memory allocated to the Control Domain of the host, the additional memory can be explicitly assigned for PVS-Accelerator.

  Perform the following steps to configure cache storage in the Control Domain memory:

  1. Run the following command to find the UUID of the host to configure for caching:

     ```
     xe host-list name-label=host-name-label --minimal
     <!--NeedCopy-->
     ```

  2. Create an SR of the special type `tmpfs`:

     ```
     xe sr-create type=tmpfs name-label=MemorySR host-uuid=
         HOST_UUID device-config:uri=""
     <!--NeedCopy-->
     ```

     > **Note:**
     >
     > For SRs of the special type `tmpfs`, the value of the required parameter `name-label`

> is disregarded and a fixed name is used instead.

3. Run the following command to create the cache storage:

   ```
   xe pvs-cache-storage-create host-uuid=HOST_UUID
   pvs-site-uuid=PVS_SITE_UUID sr-uuid=SR_UUID size=1GiB
   <!--NeedCopy-->
   ```

   Where `SR_UUID` is the UUID of the SR created in step b

Complete the cache configuration in Citrix Provisioning

After configuring PVS-Accelerator in XenServer, perform the following steps to complete the cache configuration for the Citrix Provisioning site.

In the Citrix Provisioning Administrator Console, use the Citrix Virtual Desktops Setup Wizard or the Streaming VM Wizard (depending on your deployment type) to access the Proxy capability. Although both wizards are similar and share many of the same screens, the following differences exist:

- The **Citrix Virtual Desktops Setup Wizard** is used to configure VMs running on XenServer hypervisor that is controlled using Citrix Virtual Desktops.
- The **Streaming VM Wizard** is used to create VMs on a host. It does not involve Citrix Virtual Desktops.

Launch the Citrix Provisioning Administrator Console:

1. Navigate to the Citrix Provisioning site.
2. Select the Citrix Provisioning site, right-click to expose a contextual menu.
3. Choose the appropriate wizard based on the deployment. Select the option **Enable PVS-Accelerator for all Virtual Machines** to enable the PVS-Accelerator feature.
4. If you are enabling virtual disk caching for the first time, the **XenServer** screen appears on the Streamed Virtual Machine Setup wizard. It displays the list of all Citrix Provisioning sites configured on XenServer that have not yet been associated with a Citrix Provisioning site. Using the list, select a Citrix Provisioning site to apply PVS-Accelerator. This screen is not displayed when you run the wizard for the same Citrix Provisioning site using the same XenServer host.
5. Click **Next** to complete the caching configuration.
6. Click **Finish** to provision Citrix Virtual Desktops or Streamed VMs and associate the selected Citrix Provisioning site with the PVS Accelerator in XenServer. When this step is complete, the **View PVS Servers** button in the **PVS-Accelerator configuration** window is enabled in XenCenter. Clicking the **View PVS Servers** button displays the IP addresses of all PVS Servers associated with the Citrix Provisioning site.

Caching operation

The PVS-Accelerator functionality caches:

- **Reads** from virtual disks but not writes or reads from a write cache
- **Based on image versions**. Multiple VMs share cached blocks when they use the same image version
- Devices with any **non-persistent** write cache type
- Virtual disks with the **access mode Standard Image**. It does not work for virtual disks with the access mode Private Image
- Devices that are marked as **type Production or Test**. Devices marked as type Maintenance are not cached

PVS-Accelerator CLI operations

The following section describes the operations that customers can perform when using PVS-Accelerator using the CLI. Customers can also perform these operations using XenCenter. For more information, see PVS-Accelerator in the XenCenter documentation.

View Citrix Provisioning server addresses and ports configured by Citrix Provisioning

PVS-Accelerator works by optimizing the network traffic between a VM and the Citrix Provisioning server. When completing the configuration on the Citrix Provisioning server, the Citrix Provisioning server populates the `pvs-server` objects on XenServer with their IPs and ports. PVS-Accelerator later uses this information to optimize specifically the traffic between a VM and its Citrix Provisioning servers. The configured Citrix Provisioning servers can be listed using the following command:

```
xe pvs-server-list pvs-site-uuid=PVS_SITE_UUID params=all
<!--NeedCopy-->
```

Configure a VM for caching

PVS-Accelerator can be enabled for the VM by using any of the following tools:

- Citrix Provisioning CLI
- Citrix Virtual Desktops Setup Wizard
- Streamed VM Setup Wizard
- XenCenter
- The xe CLI

The xe CLI configures PVS-Accelerator by using the VIF of a VM. It creates a Citrix Provisioning proxy that links the VM's VIF with a Citrix Provisioning site.

To configure a VM:

1. Find the first VIF of the VM to enable caching on it:

   ```
   VIF_UUID=$(xe vif-list vm-name-label=pvsdevice_1 device=0 --
       minimal)
   <!--NeedCopy-->
   ```

2. Create the Citrix Provisioning proxy

   ```
   xe pvs-proxy-create pvs-site-uuid=PVS_SITE_UUID vif-uuid=$VIF_UUID
   <!--NeedCopy-->
   ```

Disable caching for a VM

PVS-Accelerator can be disabled for a VM by destroying the Citrix Provisioning proxy that links the VM's VIF with a `pvs-site`.

1. Find the first VIF of the VM:

   ```
   VIF_UUID=$(xe vif-list vm-name-label=pvsdevice_1 device=0 --
       minimal)
   <!--NeedCopy-->
   ```

2. Find the Citrix Provisioning proxy of the VM:

   ```
   PVS_PROXY_UUID=$(xe pvs-proxy-list vif-uuid=$VIF_UUID --minimal)
   <!--NeedCopy-->
   ```

3. Destroy the Citrix Provisioning proxy:

   ```
   xe pvs-proxy-destroy uuid=$PVS_PROXY_UUID
   <!--NeedCopy-->
   ```

Remove the PVS-Accelerator storage for a host or a site

To remove the PVS-Accelerator storage for a host or a site:

1. Find the host for which you would like to destroy the storage:

   ```
   HOST_UUID=$(xe host-list name-label=HOST_NAME --minimal)
   <!--NeedCopy-->
   ```

2. Find the UUID of the object:

```
PVS_CACHE_STORAGE_UUID=$(xe pvs-cache-storage-list host-uuid=
    $HOST_UUID --minimal)
<!--NeedCopy-->
```

3. Destroy the object:

```
xe pvs-cache-storage-destroy uuid=$PVS_CACHE_STORAGE_UUID
<!--NeedCopy-->
```

Forget the PVS-Accelerator configuration for a site

To forget the PVS-Accelerator configuration for a site:

1. Find the Citrix Provisioning site:

```
PVS_SITE_UUID=$(xe pvs-site-list name-label=My PVS Site)
<!--NeedCopy-->
```

2. Run the following command to forget the Citrix Provisioning site:

```
xe pvs-site-forget uuid=$PVS_SITE_UUID
<!--NeedCopy-->
```

Hosts and resource pools

March 5, 2024

This section 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 several simple VM management examples are discussed. It also contains procedures for dealing with physical node failures.

XenServer hosts and resource pools overview

A *resource pool* comprises multiple XenServer host installations, bound together to a single managed entity which can host Virtual Machines. If combined with shared storage, a resource pool enables VMs to be started on *any* XenServer host which has sufficient memory. The VMs can then be dynamically moved among XenServer hosts while running with a minimal downtime (live migration). If an individual XenServer host suffers a hardware failure, the administrator can restart failed VMs on another XenServer host in the same resource pool. When high availability is enabled on the resource pool, VMs automatically move to another host when their host fails. Up to 64 hosts are supported per resource pool, although this restriction is not enforced.

A pool always has at least one physical node, known as the *pool coordinator* (formerly "pool master"). The coordinator node exposes an administration interface (used by XenCenter and the XenServer command line interface, known as the xe CLI). The coordinator forwards commands to individual members as necessary.

> **Note:**
>
> When the pool coordinator fails, coordinator re-election takes place only if high availability is enabled.

Requirements for creating resource pools

A resource pool is a homogeneous (or heterogeneous with restrictions) aggregate of one or more XenServer hosts, up to a maximum of 64. The definition of homogeneous is:

- CPUs on the host joining the pool are the same (in terms of the vendor, model, and features) as the CPUs on hosts already in the pool.
- The host joining the pool is running the same version of XenServer software, at the same patch level, as the hosts already in the pool.

The software enforces extra constraints when joining a host to a pool. In particular, XenServer checks that the following conditions are true for the host joining the pool:

- The host is not a member of an existing resource pool.
- The host has no shared storage configured.
- The host is not hosting any running or suspended VMs.
- No active operations are in progress on the VMs on the host, such as a VM shutting down.
- The clock on the host is synchronized to the same time as the pool coordinator (for example, by using NTP).
- The management interface of the host is not bonded. You can configure the management interface when the host successfully joins the pool.
- The 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 can contain different numbers of physical network interfaces and have local storage repositories of varying size. In practice, it is often difficult to obtain multiple hosts with the exact same CPUs, and so minor variations are permitted. If it is acceptable to have hosts with varying CPUs as part of the same pool, you can force the pool-joining operation by passing the `--force` parameter.

All hosts in the pool must be in the same site and connected by a low latency network.

> **Note:**
>
> Servers providing shared NFS or iSCSI storage for the pool must have a static IP address.

A pool must contain shared storage repositories to select on which XenServer host to run a VM and to move a VM between XenServer hosts dynamically. If possible create a pool after shared storage is available. We recommend that you move existing VMs with disks located in local storage to shared storage after adding shared storage. You can use the `xe vm-copy` command or use XenCenter to move VMs.

Create a resource pool

Resource pools can be created using XenCenter or the CLI. When a new host joins 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. This configuration is applied to the joining host in the pool unless you explicitly make the resources shared after the host joins the pool.
- The joining host inherits existing shared storage repositories in the pool. 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 reconfigured, includes:
  - The IP addresses of the management NICs, which are preserved from the original configuration.
  - The location of the management interface, which remains the same as the original configuration. For example, if the other pool hosts have management interfaces on a bonded interface, the joining host must be migrated to the bond after joining.
  - Dedicated storage NICs, which must be reassigned to the joining host from XenCenter or the CLI, and the PBDs replugged to route the traffic accordingly. This is because IP addresses are not assigned as part of the pool join operation, and the storage NIC works only when this is correctly configured. For more information on how to dedicate a storage NIC from the CLI, see Manage networking.

> **Note:**
>
> You can only join a new host to a resource pool when the host's management interface is on the same tagged VLAN as that of the resource pool.

Add a host to a pool by using the xe CLI

> **Note:**
>
> We recommend to update your pool and the joining host to the same level before attempting the join.

1. Open a console on the XenServer host that you want to join to a pool.
2. Join the XenServer host to the pool by issuing the command:

   ```
   xe pool-join master-address=<address of pool coordinator> master-
       username=<administrator username> master-password=<password>
   <!--NeedCopy-->
   ```

   The `master-address` must be set to the fully qualified domain name of the pool coordinator. The `password` must be the administrator password set when the pool coordinator was installed.

> **Note:**
>
> When you join a host to a pool, the administrator password for the joining host is automatically changed to match the administrator password of the pool coordinator.

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
<!--NeedCopy-->
```

Create heterogeneous resource pools

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

> **Note:**
>
> The CPUs of XenServer hosts joining heterogeneous pools must be of the same vendor (that is, AMD, Intel) as the CPUs of the hosts already in the pool. However, the hosts are not required to be the same type at the level of family, model, or stepping numbers.

XenServer simplifies the support of heterogeneous pools. Hosts can now be added to existing resource pools, irrespective of the underlying CPU type (as long as the CPU is from the same vendor family). The pool feature set is dynamically calculated every time:

- A new host joins the pool
- A pool member leaves the pool
- A pool member reconnects following a reboot

Any change in the pool feature set does not affect VMs that are currently running in the pool. A Running VM continues to use the feature set which was applied when it was started. This feature set is fixed at boot and persists across migrate, suspend, and resume operations. If the pool level drops when a less-capable host joins the pool, a running VM can be migrated to any host in the pool, except the newly added host. When you move or migrate a VM to a different host within or across pools, XenServer compares the VM's feature set against the feature set of the destination host. If the feature sets are found to be compatible, the VM is allowed to migrate. This enables the VM to move freely within and across pools, regardless of the CPU features the VM is using. If you use Workload Balancing to select an optimal destination host to migrate your VM, a host with an incompatible feature set will not be recommended as the destination host.

Add shared storage

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

To add NFS shared storage to a resource pool by 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 following command:

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

`device-config:server` is the host name of the NFS server and `device-config:serverpath` is the path on the NFS server. As `shared` is set to true, shared storage is automatically connected to every XenServer host in the pool. Any XenServer hosts that join later are also connected to the storage. The Universally Unique Identifier (UUID) of the storage repository is printed on the screen.

3. Find the UUID of the pool by running the following command:

```
xe pool-list
<!--NeedCopy-->
```

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

```
xe pool-param-set uuid=pool_uuid default-SR=sr_uuid
<!--NeedCopy-->
```

As the shared storage has been set as the pool-wide default, all future VMs have their disks created on shared storage by default. For information about creating other types of shared storage, see Storage repository formats.

Remove XenServer hosts from a resource pool

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

When you remove (*eject*) a host from a pool, the machine is rebooted, reinitialized, and left in a state similar to a fresh installation. Do not eject XenServer hosts from a pool if there is important data on the local disks.

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

1. Open a console on any host in the pool.
2. Find the UUID of the host by running the following command:

```
xe host-list
<!--NeedCopy-->
```

3. Eject the required host from the pool:

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

The XenServer host is 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 is erased when a host is ejected from the pool. If you want to preserve this data, copy the VM to shared storage on the pool using XenCenter, or the `xe vm-copy` CLI command.

When XenServer hosts containing locally stored VMs are ejected from a pool, the VMs will be present in the pool database. The locally stored VMs are also visible to the other XenServer hosts. The VMs do not start until the virtual disks associated with them have been changed to point at shared storage seen by other XenServer hosts in the pool, or removed. Therefore, we recommend that you move any local storage to shared storage when joining a pool. Moving to shared storage allows individual XenServer hosts to be ejected (or physically fail) without loss of data.

> **Note:**
>
> When a host is removed from a pool that has its management interface on a tagged VLAN network, the machine is rebooted and its management interface will be available on the same network.

Prepare a pool of XenServer hosts for maintenance

Before performing maintenance operations on a host that is part of a resource pool, you must disable it. Disabling the host prevents any VMs from being started on it. You must then migrate its VMs to another XenServer host in the pool. You can do this by placing the XenServer host in to Maintenance mode using XenCenter. For more information, see Run in maintenance mode in the XenCenter documentation.

Backup synchronization occurs every 24 hrs. Placing the pool coordinator in to maintenance mode results in the loss of the last 24 hrs of RRD updates for offline VMs.

> **Warning:**
>
> We highly recommend rebooting all XenServer hosts before installing an update and then verifying their configuration. Some configuration changes only take effect when the XenServer host is rebooted, so the reboot may uncover configuration problems that can cause the update to fail.

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

1. Run the following command:

```
xe host-disable uuid=XenServer_host_uuid
xe host-evacuate uuid=XenServer_host_uuid
<!--NeedCopy-->
```

This command disables the XenServer host and then migrates any running VMs to other XenServer hosts in the pool.

2. Perform the desired maintenance operation.
3. Enable the XenServer host when the maintenance operation is complete:

```
xe host-enable
<!--NeedCopy-->
```

4. Restart any halted VMs and resume any suspended VMs.

Export resource pool data

The Export Resource Data option allows you to generate a resource data report for your pool and export the report into an .xls or .csv file. This report provides detailed information about various resources in the pool such as hosts, networks, storage, virtual machines, VDIs, and GPUs. This feature enables administrators to track, plan, and assign resources based on various workloads such as CPU, storage, and network.

> **Note:**
>
> Export Resource Pool Data is available for XenServer Premium Edition customers.

The list of resources and various types of resource data that are included in the report:

Server:

- Name
- Pool Coordinator
- UUID
- Address
- CPU Usage
- Network (avg/max KBs)
- Used Memory
- Storage
- Uptime
- Description

Networks:

- Name
- Link Status
- MAC
- MTU

- VLAN
- Type
- Location

VDI:

- Name
- Type
- UUID
- Size
- Storage
- Description

Storage:

- Name
- Type
- UUID
- Size
- Location
- Description

VMs:

- Name
- Power State
- Running on
- Address
- MAC
- NIC
- Operating System
- Storage
- Used Memory
- CPU Usage
- UUID
- Uptime
- Template
- Description

GPU:

- Name
- Servers

- PCI Bus Path
- UUID
- Power Usage
- Temperature
- Used Memory
- Computer Utilization

> **Note:**
>
> Information about GPUs is available only if there are GPUs attached to your XenServer host.

To export resource data

1. In the XenCenter Navigation pane, select **Infrastructure** and then select the pool.
2. Select the **Pool** menu and then **Export Resource Data**.
3. Browse to a location where you would like to save the report and then click **Save**.

Host power-on

Powering on hosts remotely

You can use the XenServer host Power On feature to turn a host on and off remotely, either from XenCenter or by using the CLI.

To enable host power, the host 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 the RACADM command-line utility on the host with the remote access controller and enabling DRAC and its interface. RACADM is often included in the DRAC management software. For more information, see Dell's DRAC documentation.
- A custom script based on the management API that enables you to turn the power on and off through XenServer. For more information, see *Configuring a custom script for the Host Power On feature* in the following section.

Using the Host Power On feature requires two tasks:

1. Ensure the hosts in the pool support controlling the power remotely. For example, they have Wake on LAN functionality or a DRAC card, or you have created a custom script.
2. Enable the Host Power On functionality using the CLI or XenCenter.

Use the CLI to manage host power-on

You can manage the Host Power On feature using either the CLI or XenCenter. This section 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 on hosts using either the CLI or XenCenter.

To enable host power-on by using the CLI

Run the command:

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

For DRAC the keys are `power_on_ip` to specify the password if you are using the secret feature. For more information, see Secrets.

To turn on hosts remotely by using the CLI

Run the command:

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

Configure a custom script for the Host Power On feature

If your host's 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 create custom scripts for DRAC and Wake on LAN remote-power solutions.

This section 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 a XenServer host. Alternatively, you can specify it in XenCenter and use the XenCenter UI features to interact with it.

The XenServer API is documented in the XenServer Management API.

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

Key/Value Pairs To use Host Power On, configure the `host.power_on_mode` and `host.power_on_config` keys. See the following section for information about the values.

There is also an API call that lets you set these fields simultaneously:

```
void host.set_host_power_on_mode(string mode, Dictionary<string,string>
    config)
<!--NeedCopy-->
```

host.power_on_mode

- **Definition**: 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.
  - "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**: Contains key/value pairs for mode configuration. Provides additional information for DRAC.
- **Possible values:**
  - If you configured DRAC as the type of remote-power solution, you must also specify one of the following keys:
    * "power_on_ip": The IP address you specified configured to communicate with the power-control card. Alternatively, you can type the domain name for the network interface where DRAC is configured.
    * "power_on_user": The DRAC user name associated with the management processor, which you may 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" . For more information, see Secrets.

- **Type**: Map (string, string)

Sample script The 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` in all custom scripts.

The result appears 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
<!--NeedCopy-->
```

> **Note:**
>
> After creating the script, save it in /etc/xapi.d/plugins with a .py extension.

Communicate with XenServer hosts and resource pools

TLS

XenServer uses the TLS 1.2 protocol to encrypt management API traffic. Any communication between XenServer and management API clients (or appliances) uses the TLS 1.2 protocol.

> **Important:**
>
> We do not support customer modifications to the cryptographic functionality of the product.

XenServer uses the following cipher suite:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

In addition, the following cipher suites are also supported for backwards compatibility with some versions of Citrix Virtual Apps and Desktops:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_CBC_SHA256

SSH

When using an SSH client to connect directly to the XenServer host the following algorithms can be used:

Ciphers:

- aes128-ctr
- aes256-ctr
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com

MACs:

- hmac-sha2-256
- hmac-sha2-512
- hmac-sha1

KexAlgorithms:

- curve25519-sha256
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521
- diffie-hellman-group14-sha1

HostKeyAlgorithms:

- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384
- ecdsa-sha2-nistp521
- ssh-ed25519
- ssh-rsa

If you want to disable SSH access to your XenServer host, you can do this in `xsconsole`.

1. From XenCenter, open the host console and log in as `root`.
2. Type `xsconsole`.
3. In `xsconsole`, go to **Remote Service Configuration** > **Enable/Disable Remote Shell**.

   The console displays whether remote shell is enabled.

4. To change whether the remote shell is enabled or disabled, press **Enter**.

> **Important:**
>
> We do not support customer modifications to the cryptographic functionality of the product.

Install a TLS certificate on your host

The XenServer host comes installed with a default TLS certificate. However, to use HTTPS to secure communication between XenServer and Citrix Virtual Apps and Desktops, install a certificate provided by a trusted certificate authority.

This section describes how to install certificates by using the xe CLI. For information about working with certificates by using XenCenter, see the XenCenter documentation.

Ensure that your TLS certificate and its key meet the following requirements:

- The certificate and key pair are an RSA key.
- The key matches the certificate.
- The key is provided in a separate file to the certificate.
- The certificate is provided in a separate file to any intermediate certificates.
- The key file must be one of the following types: `.pem` or `.key`.
- Any certificate files must be one of the following types: `.pem`, `.cer`, or `.crt`.
- The key is greater than or equal to 2048 bits and less than or equal to 4096 bits in length.
- The key is an unencrypted PKCS #8 key and does not have a passkey.
- The key and certificate are in base-64 encoded 'PEM'format.
- The certificate is valid and has not expired.
- The signature algorithm is SHA-2 (SHA256).

The xe CLI warns you when the certificate and key you choose do not meet these requirements.

Where do I get a TLS certificate?

- You might already have a trusted certificate that you want to install on your XenServer host.
- Alternatively, you can create a certificate on your server and send it to your preferred certificate authority to be signed. This method is more secure as the private key can remain on the XenServer host and not be copied between systems.

  Creating a TLS certificate has the following steps:

  1. Generate a certificate signing request
  2. Send the certificate signing request to a certificate authority
  3. Install the signed certificate on your XenServer host

1. Generate a certificate signing request First, generate a private key and certificate signing request. On the XenServer host, complete the following steps:

1. To create a private key file, run the following command:

   ```
   openssl genrsa -des3 -out privatekey.pem 2048
   <!--NeedCopy-->
   ```

   You are prompted for a pass phrase. This pass phrase is removed in a following step.

2. Remove the pass phrase from the key:

   ```
   openssl rsa -in privatekey.pem -out privatekey.nop.pem
   <!--NeedCopy-->
   ```

3. Create the certificate signing request by using the private key:

   ```
   openssl req -new -key privatekey.nop.pem -out csr
   <!--NeedCopy-->
   ```

4. Follow the prompts to provide the information necessary to generate the certificate signing request.

   - **Country Name**. Enter the TLS Certificate country codes for your country. For example, CA for Canada or JM for Jamaica. You can find a list of TLS Certificate country codes on the web.
   - **State or Province Name (full name)**. Enter the state or province where the pool is located. For example, Massachusetts or Alberta.
   - **Locality Name**. The name of the city where the pool is located.
   - **Organization Name**. The name of your company or organization.
   - **Organizational Unit Name**. Enter the department name. This field is optional.
   - **Common Name**. Enter the FQDN of your XenServer host. We recommend specifying either an FQDN or an IP address that does not expire.
   - **Email Address**. This email address is included in the certificate when you generate it.

   The certificate signing request is saved in the current directory and is named `csr`.

5. Display the certificate signing request in the console window by running the following command:

   ```
   cat csr
   <!--NeedCopy-->
   ```

6. Copy the entire certificate signing request and use this information to request the certificate from the certificate authority.

   Example certificate signing request:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIDBDCCAewCAQAwgYsxCzAJBgNVBAYTAlVLMRcwFQYDVQQIDA5DYW1icmlkZ2Vz
aGlyZTESMBAGA1UEBwwJQ2FtYnJpZGdlMRIwEAYDVQQKDAlYZW5TZXJ2ZXIxFTAT
...
SdYCkFdo+85z8hBULFzSH6jgSP0UGQU0PcfIy7KPKyI4jnFQqeCDvLdWyhtAx9gq
Fu40qMSm1dNCFfnACRwYQkQgqCt/RHeUtl8srxyZC+odbunnV+ZyQdmLwLuQySUk
ZL8naumG3yU=
-----END CERTIFICATE REQUEST-----
<!--NeedCopy-->
```

2. Send the certificate signing request to a certificate authority Now that you have generated the certificate signing request, you can submit the request to your organization's preferred certificate authority.

A certificate authority is a trusted third-party that provides digital certificates. Some certificate authorities require the certificates to be hosted on a system that is accessible from the internet. We recommend not using a certificate authority with this requirement.

The certificate authority responds to your signing request and provides the following files:

- the signed certificate
- a root certificate
- if applicable, an intermediate certificate

You can now install all these files on your XenServer host.

3. Install the signed certificate on your XenServer host After the certificate authority reponds to the certificate signing request, complete the following steps to install the certificate on your XenServer host:

1. Get the signed certificate, root certificate and, if the certificate authority has one, the intermediate certificate from the certificate authority.
2. Copy the key and certificates to the XenServer host.
3. Run the following command on the host:

   ```
   xe host-server-certificate-install certificate=<
       path_to_certificate_file> private-key=<path_to_private_key>
       certificate-chain=<path_to_chain_file>
   ```

   The `certificate-chain` parameter is optional.

For additional security, you can delete the private key file after the certificate is installed.

Manage the administrator password

When you first install a XenServer host, you set an administrator or *root* password. You use this password to connect XenCenter to your host or (with user name `root`) to log into **xsconsole**, the system configuration console.

If you join a host to a pool, the administrator password for the host is automatically changed to match the administrator password of the pool coordinator.

> **Note:**
>
> XenServer administrator passwords must contain only ASCII characters.

Change the password

You can use XenCenter, the xe CLI, or **xsconsole** to change the administrator password.

XenCenter To change the administrator password for a pool or standalone host by using XenCenter, complete the following steps:

1. In the **Resources** pane, select the pool or any host in the pool.
2. On the **Pool** menu or on the **Server** menu, select **Change Server Password**.

To change the root password of a standalone host, select the host in the **Resources** pane, and click **Password** and then **Change** from the **Server** menu.

If XenCenter is configured to save your host login credentials between sessions, the new password is remembered. For more information, see Store your host connection state.

After changing the administrator password, rotate the pool secret. For more information, see Rotate the pool secret.

xe CLI To change the administrator password by using the xe CLI, run the following command on a host in the pool:

```
  xe user-password-change new=<new_password>
<!--NeedCopy-->
```

> **Note:**
>
> Ensure that you prefix the command with a space to avoid storing the plaintext password in the command history.

After changing the administrator password, rotate the pool secret. For more information, see Rotate the pool secret.

xsconsole To change the administrator password for a pool or a standalone host by using **xsconsole**, complete the following steps:

1. On the pool coordinator, go to the console.
2. Log in as `root`.
3. Type `xsconsole`. Press **Enter**. The **xsconsole** is displayed.
4. In **xsconsole**, use the arrow keys to navigate to the **Authentication** option. Press **Enter**.
5. Navigate to **Change Password**. Press **Enter**.
6. Authenticate with the administrator password.
7. In the **Change Password** dialog:

    a) Enter your current password.
    b) Enter a new password.
    c) Enter the new password again to confirm it.

    The **Password Change Successful** screen is displayed. Press **Enter** to dismiss.

If the host is pool coordinator, this updated password is now propagated to the other hosts in the pool.

After changing the administrator password, rotate the pool secret. For more information, see Rotate the pool secret.

Reset a lost root password

If you lose the administrator (root) password for your XenServer host, you can reset the password by accessing the host directly.

1. Reboot the XenServer host.
2. When the GRUB menu shows, press **e** to edit the boot menu entry.
3. Add `init=/sysroot/bin/sh` to the line that starts with `module2`.
4. Press **Ctrl-X** to boot into a root shell.
5. At the command shell, run the following commands:

```
chroot /sysroot
passwd

(type the new password twice)

sync
/sbin/reboot -f
<!--NeedCopy-->
```

If the host is pool coordinator, this updated password is now propagated to the other hosts in the pool.

After changing the administrator password, rotate the pool secret.

Rotate the pool secret

The pool secret is a secret shared among the hosts in a pool that enables the host to prove its membership to a pool.

Because users with the Pool Admin role can discover this secret, it is good practice to rotate the pool secret if one of these users leaves your organization or loses their Pool Admin role.

You can rotate the pool secret by using XenCenter or the xe CLI.

XenCenter

To rotate the pool secret for a pool by using XenCenter, complete the following steps:

1. In the **Resources** pane, select the pool or any host in the pool.
2. On the **Pool** menu, select **Rotate Pool Secret**.

When you rotate the pool secret, you are also prompted to change the root password. If you rotated the pool secret because you think that your environment has been compromised, ensure that you also change the root password. For more information, see Change the password.

xe CLI

To rotate the pool secret by using the xe CLI, run the following command on a host in the pool:

```
xe pool-secret-rotate
<!--NeedCopy-->
```

If you rotated the pool secret because you think that your environment has been compromised, ensure that you also change the root password. For more information, see Change the password.

Enable IGMP snooping on your XenServer pool

XenServer sends multicast traffic to all guest VMs leading to unnecessary load on host devices by requiring them to process packets they have not solicited. Enabling IGMP snooping prevents hosts on a local network from receiving traffic for a multicast group they have not explicitly joined, and improves the performance of multicast. IGMP snooping is especially useful for bandwidth-intensive IP multicast applications such as IPTV.

> **Notes:**
>
> - IGMP snooping is available only when the network back-end uses Open vSwitch.
> - When enabling this feature on a pool, it may also be necessary to enable IGMP querier on one of the physical switches. Or else, multicast in the sub network falls back to broadcast and may decrease XenServer performance.
> - When enabling this feature on a pool running IGMP v3, VM migration or network bond failover results in IGMP version switching to v2.
> - To enable this feature with a GRE network, users must set up an IGMP Querier in the GRE network. Alternatively, you can forward the IGMP query message from the physical network into the GRE network. Or else, multicast traffic in the GRE network can be blocked.

You can enable IGMP snooping on a pool by using XenCenter or the xe CLI.

XenCenter

1. Navigate to **Pool Properties**.
2. Select **Network Options**. Here you can enable or disable IGMP snooping.

xe CLI

1. Get the pool UUID:

   ```
   xe pool-list
   ```

2. Enable/disable IGMP snooping for the pool:

   ```
   xe pool-param-set [uuid=pool-uuid] [igmp-snooping-enabled=true|
   false]
   ```

After enabling IGMP snooping, you can view the IGMP snooping table using the xe CLI.

View the IGMP snooping table

Use the following command to view the IGMP snooping table:

```
ovs-appctl mdb/show [bridge name]
```

> **Note:**
>
> You can get the bridge name using `xe network-list`. These bridge names can be `xenbr0`,

> `xenbr1`, `xenapi`, or `xapi0`.

This outputs a table with four columns:

- port: The port of the switch (OVS).
- VLAN: The VLAN ID of the traffic.
- GROUP: The multicast group that the port solicited.
- Age: The age of this record in seconds.

If the **GROUP** is a multicast group address, this means an IGMP Report message was received on the associated switch port. This means that a receiver (member) of the multicast group is listening on this port.

Take the following example which contains two records:

port	VLAN	GROUP	Age
14	0	227.0.0.1	15
1	0	querier	24

The first record shows that there is a receiver listening on port 14 for the multicast group 227.0.0.1. The Open vSwitch forwards traffic destined for the 227.0.0.1 multicast group to listening ports for this group only (in this example, port 14), rather than broadcasting to all ports. The record linking port 14 and group 227.0.0.1 was created 15 seconds ago. By default, the timeout interval is 300 seconds. This means that if the switch does not receive any further IGMP Report messages on port 14 for 300 seconds after adding the record, the record expires and is removed from the table.

In the second record, the **GROUP** is **querier**, meaning that IGMP Query messages have been received on the associated port. A querier periodically sends IGMP Query messages, which are broadcasted to all switch ports, to determine which network nodes are listening on a multicast group. Upon receiving an IGMP Query message, the receiver responds with an IGMP Report message, which causes the receiver's multicast record to refresh and avoid expiration.

The **VLAN** column indicates to the VLAN that a receiver/querier lives. '0'means native VLAN. If you want to run multicast on some tagged VLAN, ensure that there are records on the VLAN.

> **Note:**
>
> For the VLAN scenario, you should have a querier record with a VLAN column value equal to the VLAN ID of the network, otherwise multicast won't work in the VLAN network.

Enable migration stream compression on your XenServer pool

During the live migration of a VM, its memory is transferred as a data stream between two hosts using the network. The migration stream compression feature compresses this data stream, speeding up the memory transfer on slow networks. This feature is disabled by default, but this can be changed by using XenCenter or the xe CLI. For more information, see Pool Properties - Advanced and Pool parameters. Alternatively, you can enable compression when migrating a VM by using the command line. For more information, see the `vm-migrate` command in VM Commands.

Certificate verification

April 16, 2024

When certificate verification is enabled for a pool, all TLS communication endpoints on its management network use certificates to validate the identity of their peers before transmitting confidential information.

Behavior

Connections initiated by a XenServer host on the management network require the destination endpoint to provide a TLS certificate to verify its identity. This requirement affects the following items that are part of the pool or interact with the pool:

- Hosts in the pool
- XenCenter
- Third-party clients that use the API

Certificate verification is compatible with both the self-signed certificates provided by XenServer and user-installed certificates signed by a trusted authority. For more information, see Install a TLS certificate on your host.

Each XenServer host in a pool has two certificates that identify it:

- *Pool-internal identity certificates* are used to secure communications between hosts within the pool. For communication within the pool, XenServer always uses self-signed certificates.
- *Server identity certificates* are used to verify the identity of a XenServer host to any client applications that communicate with the pool on the management network. For communication between the host and a client application, you can use self-signed certificates or you can install your own TLS certificates on your hosts.

When a host first joins the pool or a client first makes a connection to the pool, the pool trusts the connection. During this first connection, certificates are exchanged between the pool and the joining host or the connecting client. For all subsequent communications by this host or client on the management network, the certificates are used to verify the identity of the parties involved in the communication.

We recommend that you enable certificate verification on all your hosts and pools. For a XenServer host to successfully join a pool, both the host and the pool must have certificate verification either enabled or disabled. If certificate verification is enabled on one and not the other, the join operation is not successful. XenCenter provides a warning message that advises you to enable certificate verification on the pool or on the joining host.

When a host leaves a pool with certificate verification enabled, both the host and the pool delete the certificates that relate to the other.

The Workload Balancing virtual appliance can be used with certificate verification. You must ensure that the Workload Balancing self-signed certificates are installed into your XenServer host.

The XenServer Conversion Manager virtual appliance does not connect to XenServer hosts and so is exempt from the certification checking requirement when it acts as a TLS client end point.

Enabling certificate verification for your pool

Certificate verification is enabled by default on fresh installations of XenServer 8 and later. If you upgrade from an earlier version of XenServer or Citrix Hypervisor, certificate verification is not enabled automatically and you must enable it. XenCenter prompts you to enable certificate verification the next time you connect to the upgraded pool.

Before enabling certificate verification on a pool, ensure that no operations are running in the pool.

Enable by using XenCenter

XenCenter provides several ways to enable certificate verification.

- When first connecting XenCenter to a pool without certificate verification enabled, you are prompted to enable it. Click **Yes, Enable certificate verification**.
- In the **Pool** menu, select **Enable Certificate Verification**.
- On the **General** tab of the pool, right-click the entry **Certificate Verification** and choose **Enable Certificate Verification** from the menu.

Enable by using the xe CLI

To enable certificate verification for a pool, run the following command in the console of a host in the pool:

```
xe pool-enable-tls-verification
```

Managing certificates

You can install, view information about, and reset the certificates that are used to verify the identity of a host.

Installing certificates

You can install your own TLS certificate for the host to present as its identity certificate when receiving connections from client applications on the management network.

For more information, see Install a TLS certificate on your host.

Viewing certificate information

To find out whether a pool has certificate verification enabled:

- In XenCenter, look in the **General** tab for the pool. The **General** section has an entry for **Certificate Verification** which shows whether certificate verification is enabled or disabled. This tab also contains a **Certificates** section that lists the name, validity, and thumbprint for the CA certificates.
- With the xe CLI, you can run the following command:

  ```
  xe pool-param-get uuid=<pool_uuid> param-name=tls-verification-
      enabled
  ```

  If certificate verification is enabled, the line `tls-verification-enabled ( RO): true` appears in the command output.

To view information about the certificates on a XenServer host:

- In XenCenter, go to the **General** tab for that host. The **Certificates** section shows the thumbprint and the validity dates for the server identity certificate and the pool-internal identity certificate.
- With the xe CLI, you can run the following command:

  ```
  xe certificate-list
  ```

Refreshing pool-internal identity certificates

You can refresh the pool-internal identity certificate by using the xe CLI:

1. Find the UUID of the host whose certificate you want to reset by running the following command:

   ```
   xe host-list
   ```

2. To reset the certificate, run the following command:

   ```
   xe host-refresh-server-certificate host=<host_uuid>
   ```

   > **Note:**
   >
   > Any host selector parameter can be used with this command to indicate the host to reset the certificate on.

Resetting server identity certificates

You can reset the server identity certificate from XenCenter or the xe CLI. Resetting a certificate deletes the certificate from the host and installs a new self-signed certificate in its place.

To reset a certificate in XenCenter:

1. Go to the **General** tab for the host.
2. In the **Certificates** section, right-click on the certificate you want to reset.
3. From the menu, select **Reset Certificate**.
4. In the dialog that appears, click **Yes** to confirm the certificate reset.

Alternatively, in the **Server** menu, you can go to **Certificates > Reset Certificate**.

When you reset a certificate, any existing connections to the XenServer host are disconnected —including the connection between XenCenter and the host. XenCenter reconnects automatically to the host after a certificate reset.

To reset a certificate by using the xe CLI:

1. Find the UUID of the host whose certificate you want to reset by running the following command:

   ```
   xe host-list
   ```

2. To reset the certificate, run the following command:

   ```
   xe host-reset-server-certificate host=<host_uuid>
   ```

> **Note:**
>
> Any host selector parameter can be used with this command to indicate the XenServer host to reset the certificate on.

When you reset a certificate, any existing connections to the XenServer host are disconnected —including the connection between XenCenter and the host. XenCenter reconnects automatically to the host after a certificate reset.

Expiry alerts

XenCenter shows alerts in the **Notifications** view when your server identity certificates, pool-internal identity certificates, or pool CA certificates are close to their expiry date.

Temporarily disabling certificate verification

We do not recommend that you disable certificate verification after it has been enabled on a host or pool. However, XenServer provides commands that can be used to disable certificate verification on a per host basis when troubleshooting problems with certificates.

To temporarily disable certificate verification, run the following command on the host console:

```
xe host-emergency-disable-tls-verification
```

XenCenter shows an alert in the **Notifications** view when certificate verification is disabled on a host in a pool where the feature is enabled.

After you have resolved any issues with certificates on the host, ensure that you enable certificate verification on it again. To enable certificate verification again, run the following command on the host console:

```
xe host-emergency-reenable-tls-verification
```

Clustered pools

March 14, 2024

Clustering provides extra features that are required for resource pools that use GFS2 SRs. For more information about GFS2, see Configure storage.

A cluster is a pool of up to 16 XenServer hosts that are more closely connected and coordinated than hosts in non-clustered pools. The hosts in the cluster maintain constant communication with each other on a selected network. All hosts in the cluster are aware of the state of every host in the cluster. This host coordination enables the cluster to control access to the contents of the GFS2 SR.

> **Note:**
>
> The clustering feature only benefits pools that contain a GFS2 SR. If your pool does not contain a GFS2 SR, do not enable clustering in your pool.

Quorum

Each host in a cluster must always be in communication with the majority of hosts in the cluster (including itself). This state is known as a host having quorum. If a host does not have quorum, that host self-fences.

The number of hosts that must be in communication to initially achieve quorum can be different to the number of hosts a cluster requires to keep quorum.

The following table summarizes this behavior. The value of n is the total number of hosts in the clustered pool.

	Number of hosts required to achieve quorum	Number of hosts required to remain quorate
Odd number of hosts in the pool	(n+1)/2	(n+1)/2
Even number of hosts in the pool	(n/2)+1	n/2

Odd-numbered pools

To achieve the quorum value for an odd-numbered pool you require half of one more than the total number of hosts in the cluster: (n+1)/2. This is also the minimum number of hosts that must remain contactable for the pool to remain quorate.

For example, in a 5-host clustered pool, 3 hosts must be contactable for the cluster to both become active and remain quorate [(5+1)/2 = 3].

Where possible it is recommended to use an odd number of hosts in a clustered pool as this ensures that hosts are always able to determine if they have a quorate set.

Even-numbered pools

When an even-numbered clustered pool powers up from a cold start, (n/2)+1 hosts must be available before the hosts have quorum. After the hosts have quorum, the cluster becomes active.

However, an active even-numbered pool can remain quorate if the number of contactable hosts is at least n/2. As a result, it is possible for a running cluster with an even number of hosts to split exactly in half. The running cluster decides which half of the cluster self-fences and which half of the cluster has quorum. The half of the cluster that contains the node with the lowest ID that was seen as active before the cluster split remains active and the other half of the cluster self-fences.

For example, in a 4-host clustered pool, 3 hosts must be contactable for the cluster to become active [4/2 + 1 = 3]. After the cluster is active, to remain quorate, only 2 hosts must be contactable [4/2 = 2] and that set of hosts must include the host with the lowest node ID known to be active.

Self-fencing

If a host detects that it does not have quorum, it self-fences within a few seconds. When a host self-fences, it restarts immediately. All VMs running on the host are immediately stopped because the host does a hard shutdown. In a clustered pool that uses high availability, XenServer restarts the VMs according to their restart configuration on other pool members. The host that self-fenced restarts and attempts to rejoin the cluster.

If the number of live hosts in the cluster becomes less than the quorum value, all the remaining hosts lose quorum.

In an ideal scenario, your clustered pool always has more live hosts than are required for quorum and XenServer never fences. To make this scenario more likely, consider the following recommendations when setting up your clustered pool:

- Ensure that you have good hardware redundancy.
- Use a dedicated bonded network for the cluster network. Ensure that the bonded NICs are on the same L2 segment. For more information, see Networking.
- Configure storage multipathing between the pool and the GFS2 SR. For more information, see Storage multipathing.

Create a clustered pool

Before you begin, ensure the following prerequisites are met:

- All XenServer hosts in the clustered pool must have at least 2 GiB of control domain memory.

  Depending on your environment, your hosts might require more control domain memory than this. If you have insufficient control domain memory on your hosts, your pool can experience network instabililty. Network instability can cause problems for a clustered pool with GFS2 SRs. For information about changing the amount of control domain memory and monitoring the memory behavior, see Memory usage.

- All hosts in the cluster must use static IP addresses for the cluster network.
- We recommend that you use clustering only in pools containing at least three hosts, as pools of two hosts are sensitive to self-fencing the entire pool.
- Clustered pools only support up to 16 hosts per pool.
- If you have a firewall between the hosts in your pool, ensure that hosts can communicate on the cluster network using the following ports:
  - TCP: 8892, 8896, 21064
  - UDP: 5404, 5405

  For more information, see Communication ports used by XenServer.
- If you are clustering an existing pool, ensure that high availability is disabled. You can enable high availability again after clustering is enabled.
- We strongly recommend that you use a bonded network for your clustered pool that is not used for any other traffic.

If you prefer, you can set up clustering on your pool by using XenCenter. For more information, see the XenCenter product documentation.

To use the xe CLI to create a clustered pool:

1. Create a bonded network to use as the clustering network.

   > **Note:**
   >
   > We strongly recommend that you use a dedicated bonded network for your clustered pool. Do not use this network for any other traffic.

   On the XenServer host that you want to be the pool coordinator, complete the following steps:

   a) Open a console on the XenServer host.

   b) Create a network for use with the bonded NIC by using the following command:

   ```
   xe network-create name-label=bond0
   <!--NeedCopy-->
   ```

   The UUID of the new network is returned.

   c) Find the UUIDs of the PIFs to use in the bond by using the following command:

   ```
   xe pif-list
   <!--NeedCopy-->
   ```

   d) Create your bonded network in either active-active mode, active-passive mode, or LACP bond mode. Depending on the bond mode you want to use, complete one of the following actions:

- 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>,<pif_uuid_3>,<
           pif_uuid_4>
  <!--NeedCopy-->
  ```

  Type two UUIDs when you are bonding two NICs and four UUIDs when you are bonding four NICs. The UUID for the bond is returned after running the command.

- To configure the bond in active-passive or LACP bond mode, use the same syntax, add the optional `mode` parameter, and specify `lacp` or `active-backup`:

  ```
  xe bond-create network-uuid=<network_uuid> pif-uuids=<
     pif_uuid_1>, /
       <pif_uuid_2>,<pif_uuid_3>,<pif_uuid_4> /
       mode=balance-slb | active-backup | lacp
  <!--NeedCopy-->
  ```

After you have created your bonded network on the pool coordinator, when you join other XenServer hosts to the pool, the network and bond information is automatically replicated to the joining server.

For more information, see Networking.

2. Create a resource pool of at least three XenServer hosts.

   Repeat the following steps on each XenServer host that is a (non-master) pool member:

   a) Open a console on the XenServer host.

   b) Join the XenServer host to the pool on the pool coordinator by using the following command:

      ```
      xe pool-join master-address=master_address master-username=
          administrators_username master-password=password
      <!--NeedCopy-->
      ```

      The value of the `master-address` parameter must be set to the fully qualified domain name of the XenServer host that is the pool coordinator. The `password` must be the administrator password set when the pool coordinator was installed.

   For more information, see Hosts and resource pools.

3. For every PIF that belongs to this network, set `disallow-unplug=true`.

   a) Find the UUIDs of the PIFs that belong to the network by using the following command:

```
xe pif-list
<!--NeedCopy-->
```

b) Run the following command on a XenServer host in your resource pool:

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

4. Enable clustering on your pool. Run the following command on a XenServer host in your resource pool:

```
xe cluster-pool-create network-uuid=<network_uuid>
<!--NeedCopy-->
```

Provide the UUID of the bonded network that you created in an earlier step.

Destroy a clustered pool

You can destroy a clustered pool. After you destroy a clustered pool, the pool continues to exist, but is no longer clustered and can no longer use GFS2 SRs.

To destroy a clustered pool, run the following command:

```
xe cluster-pool-destroy cluster-uuid=<uuid>
```

Manage your clustered pool

When managing your clustered pool, the following practices can decrease the risk of the pool losing quorum.

Add or remove a host on a clustered pool

When adding or removing a host on a clustered pool, ensure that all the hosts in the cluster are online.

You can add or remove a host on a clustered pool by using XenCenter. For more information, see Add a Server to a Pool and Remove a Server From a Pool.

You can also add or remove a host on a clustered pool by using the xe CLI. For more information, see Add a host to a pool by using the xe CLI and Remove XenServer hosts from a resource pool.

Ensure that hosts are shut down cleanly

When a host is cleanly shut down, it is temporarily removed from the cluster until it is started again. While the host is shut down, it does not count toward the quorum value of the cluster. The host absence does not cause other hosts to lose quorum.

However, if a host is forcibly or unexpectedly shut down, it is not removed from the cluster before it goes offline. This host does count toward the quorum value of the cluster. Its shutdown can cause other hosts to lose quorum.

If it is necessary to shut down a host forcibly, first check how many live hosts are in the cluster. You can do this with the command `corosync-quorumtool`. In the command output, the number of live hosts is the value of `Total votes:` and the number of live hosts required to retain quorum is the value of `Quorum:`.

- If the number of live hosts is the same as the number of hosts needed to remain quorate, do not forcibly shut down the host. Doing so causes the whole cluster to fence.

  Instead, attempt to recover other hosts and increase the live hosts number before forcibly shutting down the host.

- If the number of live hosts is close to the number of hosts needed to remain quorate, you can forcibly shut down the host. However, this makes the cluster more vulnerable to fully fencing if other hosts in the pool have issues.

Always try to restart the shut down host as soon as possible to increase the resiliency of your cluster.

Use maintenance mode

Before doing something on a host that might cause that host to lose quorum, put the host into maintenance mode. When a host is in maintenance mode, running VMs are migrated off it to another host in the pool. Also, if that host was the pool coordinator, that role is passed to a different host in the pool. If your actions cause a host in maintenance mode to self-fence, you don't lose any VMs or lose your XenCenter connection to the pool.

Hosts in maintenance mode still count towards the quorum value for the cluster.

You can only change the IP address of a host that is part of a clustered pool when that host is in maintenance mode. Changing the IP address of a host causes the host to leave the cluster. When the IP address has been successfully changed, the host rejoins the cluster. After the host rejoins the cluster, you can take it out of maintenance mode.

Recover hosts that have self-fenced or are offline

It is important to recover hosts that have self-fenced. While these cluster members are offline, they count towards the quorum number for the cluster and decrease the number of cluster members that are contactable. This situation increases the risk of a subsequent host failure causing the cluster to lose quorum and shut down completely.

Having offline hosts in your cluster also prevents you from performing certain actions. In a clustered pool, every member of the pool must agree to every change of pool membership before the change can be successful. If a cluster member is not contactable, XenServer prevents operations that change cluster membership (such as host add or host remove).

Mark hosts as unrecoverable

If one or more offline hosts cannot be recovered, you can tell the clustered pool to forget them. These hosts are permanently removed from the pool. After hosts are removed from the clustered pool, they no longer count towards the quorum value.

To mark a host as unrecoverable, use the following command:

```
xe host-forget uuid=<host_uuid>
```

Recover a forgotten host

After a clustered pool is told to forget a host, the host cannot be added back into the pool.

To rejoin the clustered pool, you must reinstall XenServer on the host so that it appears as a new host to the pool. You can then join the host to the clustered pool in the usual way.

Troubleshoot your clustered pool

If you encounter issues with your clustered pool, see Troubleshoot clustered pools.

Constraints

- Clustered pools only support up to 16 hosts per pool.
- To enable HA on your clustered pool, the heartbeat SR must be a GFS2 SR.
- For cluster traffic, we strongly recommend that you use a bonded network that uses at least two different network switches. Do not use this network for any other purposes.
- Changing the IP address of the cluster network by using XenCenter requires clustering and GFS2 to be temporarily disabled.

- Do not change the bonding of your clustering network while the cluster is live and has running VMs. This action can cause hosts in the cluster to hard restart (fence).
- If you have an IP address conflict (multiple hosts having the same IP address) on your clustering network involving at least one host with clustering enabled, the cluster does not form correctly and the hosts are unable to fence when required. To fix this issue, resolve the IP address conflict.

Troubleshoot clustered pools

December 11, 2023

XenServer pools that use GFS2 to thin provision their shared block storage are clustered. These pools behave differently to pools that use shared file-based storage or LVM with shared block storage. As a result, there are some specific issues that might occur in XenServer clustered pools and GFS2 environments.

Use the following information to troubleshoot minor issues that might occur when using this feature.

All my hosts can ping each other, but I can't create a cluster. Why?

The clustering mechanism uses specific ports. If your hosts can't communicate on these ports (even if they can communicate on other ports), you can't enable clustering for the pool.

Ensure that the hosts in the pool can communicate on the following ports:

- TCP: 8892, 8896, 21064
- UDP: 5404, 5405 (not multicast)

If there are any firewalls or similar between the hosts in the pool, ensure that these ports are open.

If you have previously configured HA in the pool, disable the HA before enabling clustering.

Why am I getting an error when I try to join a new host to an existing clustered pool?

When clustering is enabled on a pool, every pool membership change must be agreed by every member of the cluster before it can succeed. If a cluster member isn't contactable, operations that change cluster membership (such as host add or host remove) fail.

To add your new host to the clustered pool:

1. Ensure that all of your hosts are online and can be contacted.

2. Ensure that the hosts in the pool can communicate on the following ports:

    - TCP: 8892, 8896, 21064
    - UDP: 5404, 5405 (not multicast)

3. Ensure that the joining host has an IP address allocated on the NIC that joins the cluster network of the pool.
4. Ensure that no host in the pool is offline when a new host is trying to join the clustered pool.
5. If an offline host cannot be recovered, mark it as dead to remove it from the cluster. For more information, see A host in my clustered pool is offline and I can't recover it. How do I remove the host from my cluster?

What do I do if some members of the clustered pool aren't joining the cluster automatically?

This issue might be caused by members of the clustered pool losing synchronization.

To resync the members of the clustered pool, use the following command:

```
xe cluster-pool-resync cluster-uuid=<cluster_uuid>
```

If the issue persists, you can try to reattach the GFS2 SR. You can do this task by using the xe CLI or through XenCenter.

Reattach the GFS2 SR by using the xe CLI:

1. Detach the GFS2 SR from the pool. On each host, run the xe CLI command `xe pbd-unplug uuid=<uuid_of_pbd>`.
2. Disable the clustered pool by using the command `xe cluster-pool-destroy cluster-uuid=<cluster_uuid>`

    If the preceding command is unsuccessful, you can forcibly disable a clustered pool by running `xe cluster-host-force-destroy uuid=<cluster_host>` on every host in the pool.

3. Enable the clustered pool again by using the command `xe cluster-pool-create network-uuid=<network_uuid> [cluster-stack=cluster_stack] [token-timeout=token_timeout] [token-timeout-coefficient=token_timeout_coefficien ]`
4. Reattach the GFS2 SR by running the command `xe pbd-plug uuid=<uuid_of_pbd>` on each host.

Alternatively, to use XenCenter to reattach the GFS2 SR:

1. In the pool **Storage** tab, right-click on the GFS2 SR and select **Detach…**.
2. From the toolbar, select **Pool > Properties**.
3. In the **Clustering** tab, deselect **Enable clustering**.
4. Click **OK** to apply your change.
5. From the toolbar, select **Pool > Properties**.
6. In the **Clustering** tab, select **Enable clustering** and choose the network to use for clustering.
7. Click **OK** to apply your change.
8. In the pool **Storage** tab, right-click on the GFS2 SR and select **Repair**.

How do I know if my host has self-fenced?

If your host self-fenced, it might have rejoined the cluster when it restarted. To see if a host has self-fenced and recovered, you can check the `/var/opt/xapi-clusterd/boot-times` file to see the times the host started. If there are start times in the file that you did not expect to see, the host has self-fenced.

Why is my host offline? How can I recover it?

There are many possible reasons for a host to go offline. Depending on the reason, the host can either be recovered or not.

The following reasons for a host to be offline are more common and can be addressed by recovering the host:

- Clean shutdown
- Forced shutdown
- Temporary power failure
- Reboot

The following reasons for a host to be offline are less common:

- Permanent host hardware failure
- Permanent host power supply failure
- Network partition
- Network switch failure

These issues can be addressed by replacing hardware or by marking failed hosts as dead.

A host in my clustered pool is offline and I can't recover it. How do I remove the host from my cluster?

You can tell the cluster to forget the host. This action removes the host from the cluster permanently and decreases the number of live hosts required for quorum.

To remove an unrecoverable host, use the following command:

```
xe host-forget uuid=<host_uuid>
```

This command removes the host from the cluster permanently and decreases the number of live hosts required for quorum.

> **Note:**
>
> If the host isn't offline, this command can cause data loss. You're asked to confirm that you're sure before proceeding with the command.

After a host is forgotten, it can't be added back into the cluster. To add this host back into the cluster, you must do a fresh installation of XenServer on the host.

I've repaired a host that was marked as dead. How do I add it back into my cluster?

A XenServer host that has been marked as dead can't be added back into the cluster. To add this system back into the cluster, you must do a fresh installation of XenServer. This fresh installation appears to the cluster as a new host.

What do I do if my cluster keeps losing quorum and its hosts keep fencing?

If one or more of the XenServer hosts in the cluster gets into a fence loop because of continuously losing and gaining quorum, you can boot the host with the `nocluster` kernel command-line argument. Connect to the physical or serial console of the host and edit the boot arguments in grub.

Example:

```
/boot/grub/grub.cfg
menuentry 'XenServer' {

        search --label --set root root-oyftuj
        multiboot2 /boot/xen.gz dom0_mem=4096M,max:4096M watchdog ucode
           =scan dom0_max_vcpus=1-16 crashkernel=192M,below=4G console=
           vga vga=mode-0x0311
        module2 /boot/vmlinuz-4.4-xen root=LABEL=root-oyftuj ro nolvm
           hpet=disable xencons=hvc console=hvc0 console=tty0 quiet vga
           =785 splash plymouth.ignore-serial-consoles nocluster
        module2 /boot/initrd-4.4-xen.img
}

menuentry 'XenServer (Serial)' {

        search --label --set root root-oyftuj
        multiboot2 /boot/xen.gz com1=115200,8n1 console=com1,vga
           dom0_mem=4096M,max:4096M watchdog ucode=scan dom0_max_vcpus
           =1-16 crashkernel=192M,below=4G
        module2 /boot/vmlinuz-4.4-xen root=LABEL=root-oyftuj ro nolvm
           hpet=disable console=tty0 xencons=hvc console=hvc0 nocluster
        module2 /boot/initrd-4.4-xen.img
}

<!--NeedCopy-->
```

What happens when the pool coordinator gets restarted in a clustered pool?

In most cases, the behavior when the pool coordinator is shut down or restarted in a clustered pool is the same as that when another pool member shuts down or restarts.

How the host is shut down or restarted can affect the quorum of the clustered pool. For more information about quorum, see Quorum.

The only difference in behavior depends on whether HA is enabled in your pool:

- If HA is enabled, a new coordinator is selected and general service is maintained.
- If HA is not enabled, there is no coordinator for the pool. Running VMs on the remaining hosts continue to run. Most administrative operations aren't available until the coordinator restarts.

Why has my pool vanished after a host in the clustered pool is forced to shut down?

If you shut down a host normally (not forcibly), it's temporarily removed from quorum calculations until it's turned back on. However, if you forcibly shut down a host or it loses power, that host still counts towards quorum calculations. For example, if you had a pool of 3 hosts and forcibly shut down 2 of them the remaining host fences because it no longer has quorum.

Try to always shut down hosts in a clustered pool cleanly. For more information, see Manage your clustered pool.

Why did all hosts within the clustered pool restart at the same time?

All hosts in an active cluster are considered to have lost quorum when the number of contactable hosts in the pool is less than these values:

- For a pool with an even number of hosts: n/2

- For a pool with an odd number of hosts: (n+1)/2

The letter n indicates the total number of hosts in the clustered pool. For more information about quorum, see Quorum.

In this situation, all hosts self-fence and you see all hosts restarting.

To diagnose why the pool lost quorum, the following information can be useful:

- In XenCenter, check the **Notifications** section for the time of the issue to see whether self-fencing occurred.
- On the cluster hosts, check `/var/opt/xapi-clusterd/boot-times` to see whether a reboot occurred at an unexpected time.
- In `Crit.log`, check whether any self-fencing messages are outputted.
- Review the `dlm_tool status` command output for fencing information.

  Example `dlm_tool status` output:

```
dlm_tool status

cluster nodeid 1 quorate 1 ring seq 8 8
daemon now 4281 fence_pid 0
node 1 M add 3063 rem 0 fail 0 fence 0 at 0 0
node 2 M add 3066 rem 0 fail 0 fence 0 at 0 0
<!--NeedCopy-->
```

When collecting logs for debugging, collect diagnostic information from all hosts in the cluster. In the case where a single host has self-fenced, the other hosts in the cluster are more likely to have useful information.

Collect full server status reports for the hosts in your clustered pool. For more information, see XenServer host logs.

Why can't I recover my clustered pool when I have quorum?

If you have a clustered pool with an even number of hosts, the number of hosts required to *achieve* quorum is one more than the number of hosts required to *retain* quorum. For more information about quorum, see Quorum.

If you are in an even-numbered pool and have recovered half of the hosts, you must recover one more host before you can recover the cluster.

Why do I see an `Invalid token` error when changing the cluster settings?

When updating the configuration of your cluster, you might receive the following error message about an invalid token (`"[[\"InternalError\",\"Invalid token\"]]"`).

You can resolve this issue by completing the following steps:

1. (Optional) Back up the current cluster configuration by collecting a server status report that includes the xapi-clusterd and system logs.
2. Use XenCenter to detach the GFS2 SR from the clustered pool.

   In the pool **Storage** tab, right-click on the GFS2 SR and select **Detach…**.
3. On any host in the cluster, run this command to forcibly destroy the cluster:

   ```
   xe cluster-pool-force-destroy cluster-uuid=<uuid>
   ```
4. Use XenCenter to reenable clustering on your pool.

   a) From the toolbar, select **Pool > Properties**.
   b) In the **Clustering** tab, select **Enable clustering** and choose the network to use for clustering.
   c) Click **OK** to apply your change
5. Use XenCenter to reattach the GFS2 SR to the pool

   In the pool **Storage** tab, right-click on the GFS2 SR and select **Repair**.

Manage users

February 22, 2024

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 XenServer authenticates locally.

The LSU, or root, is a special user account intended for system administration and has all permissions. In XenServer, the LSU is the default account at installation. XenServer authenticates the LSU account. LSU does not require any external authentication service. If an 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 more users by adding the Active Directory accounts through either XenCenter's Users tab or the xe CLI. If your environment does not use Active Directory, you are limited to the LSU account.

> **Note:**
>
> When you create users, XenServer does not assign newly created user accounts RBAC roles automatically. Therefore, these accounts do not have any access to the XenServer pool until you assign them a role.

These permissions are granted through roles, as discussed in the *Authenticating users with Active Directory (AD)* section.

Authenticate users with Active Directory (AD)

If you want to have multiple user accounts on a host or a pool, you must use Active Directory user accounts for authentication. AD accounts let XenServer users log on to a pool using their Windows domain credentials.

> **Note:**
>
> You can enable LDAP channel binding and LDAP signing on your AD domain controllers. For more information, see Microsoft Security Advisory.

You can configure varying levels of access for specific users by enabling Active Directory authentication, adding user accounts, and assigning 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.

Subjects control access to user accounts. A subject in XenServer maps to an entity on your Active Directory server (either a user or a group). When you enable external authentication, XenServer checks the credentials used to create a session against the local root credentials and then against the subject list. To permit access, create a subject entry for the person or group you want to grant access to. You can use XenCenter or the xe CLI to create a subject entry.

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

XenCenter Term	xe CLI Term
Users, Add users	Subjects, Add subjects

Even though XenServer is 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 you add Active Directory to XenServer, Active Directory users and groups become XenServer subjects. The subjects are referred to as users in XenCenter. Users/groups are authenticated by using Active Directory on logon when you register a subject with XenServer. Users and groups do not need to qualify their user name by using a domain name.

To qualify a user name, you must type the user name in Down-Level log on Name format, for example, `mydomain\myuser`.

> **Note:**
>
> By default, if you did not qualify the user name, XenCenter attempts to log in users to AD authentication servers using the domain to which it is 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 host 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, authentication succeeds.

When you join a domain, you enable Active Directory authentication for the pool. However, when a pool joins 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 can cause AD integration, and therefore user authentication, to fail or stop working.

Configure Active Directory authentication

XenServer supports use of Active Directory servers using Windows 2008 or later.

To authenticate Active Directory for XenServer hosts, you must use the same DNS server for both the Active Directory server (configured to allow interoperability) and the XenServer host.
In some configurations, the Active Directory server can 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. Alternatively, you can set the values in the PIF objects or use the installer when a manual static configuration is used.

We recommend enabling DHCP to assign host names. Do not assign the hostnames `localhost` or `linux` to hosts.

> **Warning:**
>
> XenServer host names must be unique throughout the XenServer deployment.

Note the following:

- XenServer labels its AD entry on the AD database using its hostname. If two XenServer hosts with the same hostname are joined to the same AD domain, the second XenServer overwrites the AD entry of the first XenServer. The overwriting occurs regardless of whether the hosts belong to the same or different pools. This can cause the AD authentication on the first XenServer to stop working.

  You can use the same host name in two XenServer hosts, as long as they join different AD domains.

- The XenServer hosts can be in different time-zones, because it is the UTC time that is compared. To ensure that synchronization is correct, you can use the same NTP servers for your XenServer pool and the Active Directory server.
- Mixed-authentication pools are not supported. You cannot have a pool where some hosts 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. Therefore, XenServer does not support communicating with Active Directory servers that do not use Kerberos.
- For external authentication using Active Directory to be successful, clocks on your XenServer hosts must be synchronized with the clocks on your Active Directory server. When XenServer joins the Active Directory domain, the synchronization is checked and authentication fails 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.
>
> A limitation in recent SSH clients means that SSH does not work for usernames that contain any of the following characters: `{ }  []|&`. Ensure that your usernames and Active Directory server names do not contain any of these characters.

When you add a host to a pool after enabling Active Directory authentication, you are prompted to configure Active Directory on the host joining the pool. When prompted for credentials on the joining host, type Active Directory credentials with sufficient privileges to add hosts to that domain.

Active Directory integration

Ensure 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
636	UDP/TCP	LDAP over SSL
3268	TCP	Global Catalog Search

For more information, see Communication Ports Used by XenServer.

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

Winbind

XenServer uses Winbind for authenticating Active Directory (AD) users with the AD server and to encrypt communications with the AD server.

Winbind does not support the following scenarios:

- Space at the beginning or end of a domain user or domain group name.
- Domain user names that contain 64 characters or more.

- Domain user names that include any of the special characters +<>"=/%@:,;\`
- Domain group names that include any of the special characters ,;\`

Configuring Winbind Configure Winbind behavior with the following configuration options, which can be included in the `/etc/xapi.conf` file:

- `winbind_machine_pwd_timeout`: The value of this option defines how often, in seconds, the machine password is rotated for this XenServer host. Define a value as an integer.

  The default value is 1209600 seconds (14 days). We recommend that you keep the default value or do not decrease the value below the default value to guarantee enough time to synchronize the new password between domain controllers.

- `winbind_kerberos_encryption_type`: The values for this option are strong, legacy, and all. The default value is all.

  - The value `all` permits the following cipher suites: `aes256-cts-hmac-sha1-96`, `aes128-cts-hmac-sha1-96`, and `arcfour-hmac-md5`
  - The value `strong` permits the following cipher suites: `aes256-cts-hmac-sha1-96` and `aes128-cts-hmac-sha1-96`
  - The value `legacy` permits the following cipher suites: `arcfour-hmac-md5`

    The legacy option is insecure and we recommend that you only use it to debug issues.

  For improved security, we recommend that you enforce AES encryption. To do this,

  1. Ensure the domain controller supports `aes256-cts-hmac-sha1-96` and `aes128-cts-hmacsha1-96`.
  2. Configure the domain controller to enable **The other domain supports Kerberos AES Encryption** in domain trust.

     For more information, see Method 3: Configure the trust to support AES128 and AES 256 encryption instead of RC4 encryption in the Microsoft documentation.
  3. Update the `winbind_kerberos_encryption_type` option to use the value `strong`.
  4. Restart the toolstack.

     Do not restart the toolstack while HA is enabled. If possible, temporarily disable HA before restarting the toolstack.

- `winbind_cache_time`: Winbind caches some domain information locally. The value of this option defines the number of seconds between each cache refresh. The default is 60 seconds.

After you update any of these configuration options, restart the toolstack.

How does XenServer manage the machine account password for AD integration?

Similarly to Windows client machines, Winbind automatically updates the machine account password. Winbind automatically updates the machine account password every 14 days or as specified by the configuration option `winbind_machine_pwd_timeout`.

Enable external authentication on a pool

External authentication using Active Directory can be configured using either XenCenter or the CLI using the following command.

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

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

If you are not using DHCP on the network used by Active Directory and your XenServer hosts, use the following approaches to set up your DNS:

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

   ```
   xe pif-param-set uuid=pif_uuid_in_the_dns_subnetwork \
      "other-config:domain=suffix1.com suffix2.com suffix3.com"
   <!--NeedCopy-->
   ```

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

   ```
   xe pif-reconfigure-ip mode=static dns=dnshost ip=ip \
     gateway=gateway netmask=netmask uuid=uuid
   <!--NeedCopy-->
   ```

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

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

> **Note:**
>
> External authentication is a per-host property. However, we recommend that you enable and disable external authentication on a per-pool basis. A per-pool setting allows XenServer to deal with failures that occur when enabling authentication on a particular host. XenServer also rolls back any changes that may be required, ensuring a consistent configuration 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.

Use XenCenter to disable Active Directory authentication, or the following xe command:

```
xe pool-disable-external-auth
<!--NeedCopy-->
```

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 want to manage user permissions in Active Directory, you can create a single group that you then add and delete users to/from. Alternatively, you can add and delete individual users from XenServer, or a combination of users and groups as appropriate for your authentication requirements. You can manage the subject list from XenCenter or using the CLI as described in the following section.

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 (user name and password) do not match, then an authentication request is made to the AD server. If the authentication is successful, the user's information is retrieved and validated against the local subject list. Access is denied if the authentication fails. Validation against the subject list succeeds 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 require host ssh access, the size of the AD group must not exceed 500 users.

To add an AD subject to XenServer:

```
xe subject-add subject-name=entity_name
<!--NeedCopy-->
```

The entity_name is the name of the user or group to which you want to grant access. You can include the domain of the entity (for example, 'xendt\user1'as opposed to 'user1') although the behavior is the same unless disambiguation is required.

Find the user's subject identifier. The identifier is the user or the group containing the user. Removing a group removes access to all users in that group, provided they are not also specified in the subject list. Use the `subject list` command to find the user's subject identifier. :

```
xe subject-list
<!--NeedCopy-->
```

This command returns a list of all users.

To apply a filter to the list, for example to find the subject identifier for a user `user1` in the `testad` domain, use the following command:

```
xe subject-list other-config:subject-name='testad\user1'
<!--NeedCopy-->
```

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
<!--NeedCopy-->
```

You can end any current session this user has already authenticated. For more information, see *Terminating all authenticated sessions using xe* and *Terminating individual user sessions using xe* in the following section. If you do not end sessions, users with revoked permissions may continue to access the system until they log out.

Run the following command to identify the list of users and groups with permission to access your XenServer host or pool:

```
xe subject-list
<!--NeedCopy-->
```

Remove access for a user

When a user is authenticated, they can access the host until they end their session, or another user ends their session. Removing a user from the subject list, or removing them from a group in the subject list, doesn't automatically revoke any already-authenticated sessions that the user has. Users can continue to access the pool using XenCenter or other API sessions that they have already created. XenCenter and the CLI provide facilities to end individual sessions, or all active sessions forcefully. See the XenCenter documentation for information on procedures using XenCenter, or the following section for procedures using the CLI.

Terminate all authenticated sessions using xe

Run the following CLI command to end all authenticated sessions using xe:

```
xe session-subject-identifier-logout-all
<!--NeedCopy-->
```

Terminate individual user sessions using xe

1. Determine the subject identifier whose session you want to log out. Use either the `session-subject-identifier-list` or `subject-list` xe commands to find the subject identifier. The first command shows users who have sessions. The second command shows all users but can be filtered. For example, by using a command like `xe subject-list other-config:subject-name=xendt\\user1`. You may need a double backslash as shown depending on your shell).
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
<!--NeedCopy-->
```

Leave an AD domain

> **Warning:**
>
> When you leave the domain, any users who authenticated to the pool or host with Active Directory credentials are disconnected.

Use XenCenter to leave an AD domain. For more information, see the XenCenter documentation. Alternately run the `pool-disable-external-auth` command, specifying the pool UUID if necessary.

> **Note:**
>
> Leaving the domain does not delete the host objects from the AD database. Refer to the Active Directory documentation for information about how to detect and remove your disabled host entries.

Role-based access control

May 17, 2023

The Role-Based Access Control (RBAC) feature in XenServer 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). The roles have associated XenServer permissions to perform certain operations.

Permissions are not assigned to users directly. Users acquire permissions through roles assigned to them. Therefore, managing individual user permissions becomes a matter of assigning the user to the appropriate role, which simplifies common operations. XenServer maintains a list of authorized users and their roles.

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

RBAC also provides an Audit Log feature for compliance and auditing.

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. The local super user is the default account at installation in XenServer. The LSU is authenticated through XenServer and not through an external authentication service. 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 through SSH.

RBAC process

The following section describes the standard process for implementing RBAC and assigning a user or group a role:

1. Join the domain. For more information, see Enabling external authentication on a pool.
2. Add an Active Directory user or group to the pool. This becomes a subject. For more information, see To add a subject to RBAC.
3. Assign (or change) the subject's RBAC role. For more information, see To assign an RBAC role to a subject.

RBAC roles and permissions

April 15, 2024

Roles

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

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

> **Note:**
>
> The local super user (root) has the "Pool Admin"role. The Pool Admin role has the same permissions as the local root.
>
> If you remove the Pool Admin role from a user, consider also changing the root password and rotating the pool secret. For more information, see Pool Security.

- *Pool Operator* (Pool Operator) –can do everything apart from adding/removing users and changing their roles. This role is focused mainly on host and pool management (that is, creating storage, making pools, managing the hosts and so on.)
- *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:**
>
> - To apply updates to XenServer 8 pools, you must be logged in to XenCenter as a Pool Administrator or Pool Operator, or using a local root account.
> - When using Active Directory groups to grant access for Pool Administrator users who require host SSH access, the number of users in the Active Directory group must not exceed 500.

For a summary of the permissions available for each role and for information on the operations available for each permission, see *Definitions of RBAC roles and permissions* in the following section.

When you create a user in XenServer, you must first assign a role to the newly created user before they can use the account. XenServer **does not** automatically assign a role to the newly created user. As a result, these accounts do not have any access to XenServer pool until you assign them a role.

1. Modify the subject to 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*.

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					
Install a TLS certificate on a server	X					
Apply updates to a pool	X	X				
Rolling Pool Upgrade	X					
Import OVF/OVA packages and disk images	X	X				
Import XVA packages	X	X	X			
Export OVF/OVA/XVA packages and disk images	X	X	X	X		
Set cores per socket	X	X	X	X		

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
Convert virtual machines using XenServer Conversion Manager	X					
Switch-port locking	X	X				
Multipathing	X	X				
Log out active user connections	X	X				
Monitor host and dom0 resources with NRPE	X					
Monitor host and dom0 resources with SNMP	X					
Create and dismiss alerts	X	X				
Cancel task of any user	X	X				
Pool management	X	X				
Live migration	X	X	X			
Storage live migration	X	X	X			

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
VM advanced operations	X	X	X			
VM create/destroy operations	X	X	X	X		
VM change CD media	X	X	X	X	X	
VM change power state	X	X	X	X	X	
View VM consoles	X	X	X	X	X	
XenCenter view management operations	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 Workload Balancing (WLB)	X	X				
Apply WLB optimization recommendations	X	X				
Accept WLB placement recommendations	X	X	X			

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
Display WLB configuration	X	X	X	X	X	X
Generate WLB reports	X	X	X	X	X	X
Connect to pool and read all pool metadata	X	X	X	X	X	X
Configure virtual GPU	X	X				
View virtual GPU configuration	X	X	X	X	X	X
Gather diagnostic information	X	X				
vCPU Hotplug	X	X	X	X		
Configure changed block tracking	X	X	X	X		
List changed blocks	X	X	X	X	X	
Configure PVS-Accelerator	X	X				
View PVS-Accelerator configuration	X	X	X	X	X	X

Role permissions	Pool Admin	Pool Operator	VM Power Admin	VM Admin	VM Operator	Read Only
Scheduled Snapshots (Add/Remove VMs to existing Snapshots Schedules)	X	X	X			
Scheduled Snapshots (Add/Modify/Delete Snapshot Schedules)	X	X				

Definitions of permissions

Assign/modify roles:

- Add and remove users
- Add and remove roles from users
- Enable and disable Active Directory integration (being joined to the domain)

This permission lets the user grant themself 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 can arbitrarily reconfigure the entire system, including RBAC.

Server backup/restore:

- Back up and restore servers
- Back up and restore pool metadata

The ability to restore a backup lets the assignee revert RBAC configuration changes.

Install a TLS certificate on a server:

This permission enables an administrator to install a TLS certificate on a server that runs Citrix Hypervisor 8.2 or later.

Apply updates to a pool:

- Synchronize your pool with the content delivery network (CDN)
- Apply the updates by migrating VMs off each host if necessary and running any necessary update tasks such as rebooting the host, restarting the toolstack, or rebooting the VMs

Rolling Pool Upgrade:

- Upgrade all hosts in a pool using the Rolling Pool Upgrade wizard.

Import OVF/OVA packages and disk images:

- Import OVF and OVA packages
- Import disk images

Import XVA packages:

- Import XVA packages

Export OVF/OVA/XVA packages and disk images:

- Export VMs as OVF/OVA packages
- Export VMs as XVA packages
- Export disk images

Set cores-per-socket:

- Set the number of cores per socket for the VM's virtual CPUs

This permission enables the user to specify the topology for the VM's virtual CPUs.

Convert VMs using XenServer Conversion Manager:

- Convert VMware ESXi/vCenter VMs to XenServer VMs

This permission lets the user convert workloads from VMware to XenServer. Convert these workloads by copying batches of VMware ESXi/vCenter VMs to the XenServer environment.

Switch-port locking:

- Control traffic on a network

This permission lets the user block all traffic on a network by default, or define specific IP addresses from which a VM can send traffic.

Multipathing:

- Enable multipathing
- Disable multipathing

Log out active user connections:

- Ability to disconnect logged in users

Monitor host and dom0 resources with NRPE:

For more information, see Monitor host and dom0 resources with NRPE.

Monitor host and dom0 resources with SNMP:

For more information, see Monitor host and dom0 resources with SNMP.

Create/dismiss alerts:

- Configure XenCenter to generate alerts when resource usage crosses certain thresholds
- Remove alerts from the Alerts view

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.

Pool management:

- Set pool properties (naming, default SRs)
- Create a clustered pool
- Enable, disable, and configure HA
- Set per-VM HA restart priorities
- Configure DR and perform DR failover, failback, and test failover operations.
- Enable, disable, and configure Workload Balancing (WLB)
- Add and remove server from pool
- Emergency transition to pool coordinator
- Emergency pool coordinator address

- Emergency recovery of pool members
- Designate new pool coordinator
- Manage pool and server certificates
- Patching
- Set server properties
- Configure server logging
- Enable and disable servers
- Shut down, reboot, and power-on servers
- Restart toolstack
- System status reports
- Apply license
- Live migration of all other VMs on a server to another server, due to either WLB, maintenance mode, or high availability
- Configure server management interfaces and secondary 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 management interface is not functioning, no logins can authenticate except local root logins.

Live migration:

- Migrate VMs from one host to another host when the VMs are on storage shared by both hosts

Storage live migration:

- Migrate from one host to another host when the VMs are not on storage shared between the two hosts
- Move Virtual Disk (VDIs) from one SR to another SR

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 host if they are not satisfied with the host XenServer selected.

VM create/destroy operations:

- Install and delete VMs
- Clone/copy VMs
- Add, remove, and configure virtual disk/CD devices
- Add, remove, and configure virtual network devices
- 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 host consoles.

Cancel own tasks:

- Enables users to cancel their own tasks

Read audit log:

- Download XenServer audit log

Configure, initialize, enable, disable WLB:

- Configure WLB
- Initialize WLB and change WLB servers
- Enable WLB
- Disable WLB

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

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 management operations:

- Create and modify global XenCenter folders
- Create and modify global XenCenter custom fields
- Create and modify global XenCenter searches

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 tasks
- View messages
- Register for and receive events

Configure virtual GPU:

- Specify a pool-wide placement policy
- Assign a virtual GPU to a VM
- Remove a virtual GPU from a VM
- Modify allowed virtual GPU types
- Create, destroy, or assign a GPU group

View virtual GPU configuration:

- View GPUs, GPU placement policies, and virtual GPU assignments.

Gather diagnostic information from XenServer:

- Initiate GC collection and heap compaction
- Gather garbage collection statistics
- Gather database statistics
- Gather network statistics

Configure changed block tracking:

- Enable changed block tracking
- Disable changed block tracking
- Destroy the data associated with a snapshot and retain the metadata
- Get the NBD connection information for a VDI
- Export a VDI over an NBD connection

Changed block tracking can be enabled only for licensed instances of XenServer Premium Edition.

List changed blocks:

- Compare two VDI snapshots and list the blocks that have changed between them.

Configure PVS-Accelerator:

- Enable PVS-Accelerator
- Disable PVS-Accelerator
- Update PVS-Accelerator cache configuration
- Add or Remove PVS-Accelerator cache configuration

View PVS-Accelerator configuration:

- View the status of PVS-Accelerator

Scheduled snapshots (Add/Remove VMs to existing Snapshots Schedules):

- Add VMs to existing snapshot schedules
- Remove VMs from existing snapshot schedules

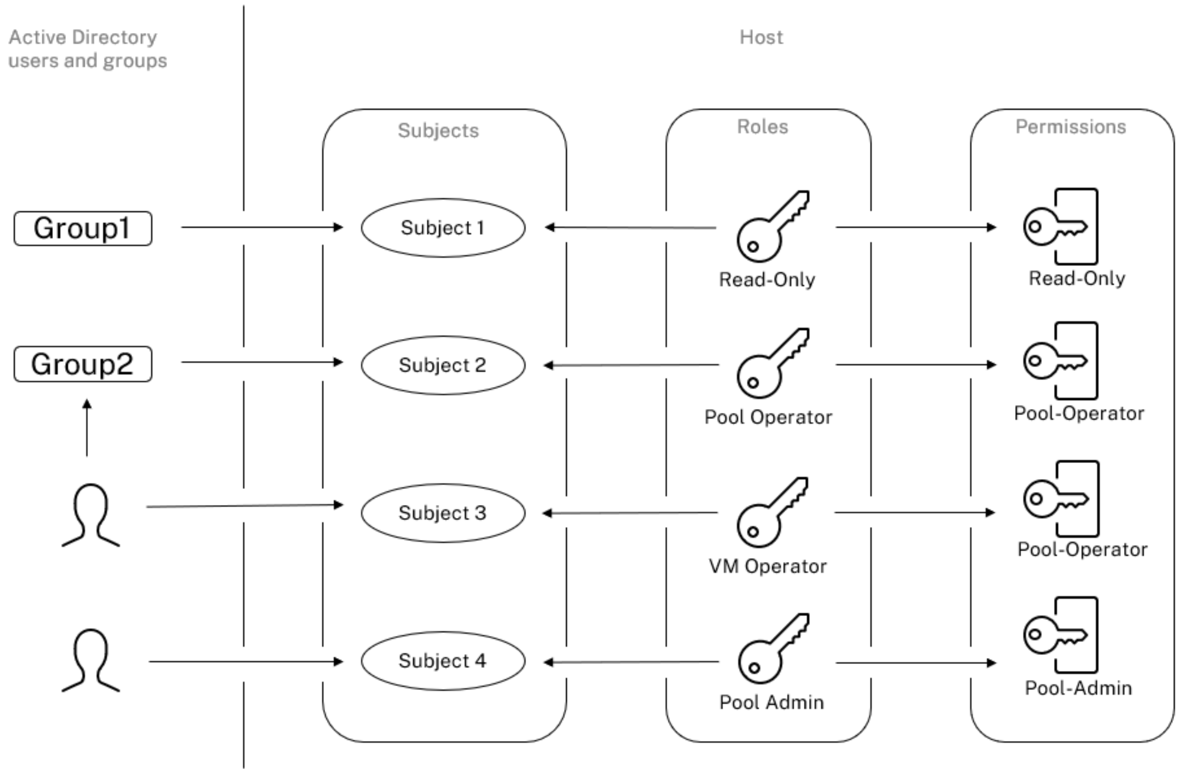
Scheduled snapshots (Add/Modify/Delete Snapshot Schedules):

- Add snapshot schedules
- Modify snapshot schedules
- Delete snapshot schedules

> **Note:**
>
> Sometimes, 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.

How does XenServer compute the roles for the session?

1. The subject is authenticated through 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 inherit all of the permissions of the associated roles.



Use RBAC with the CLI

May 17, 2023

RBAC xe CLI commands

Use the following commands to work with roles and subjects.

To list all the available defined roles

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 has full access to
        all
    features and settings, including accessing Dom0 and managing
        subjects,
    roles and external authentication

    uuid ( RO): b9ce9791-0604-50cd-0649-09b3284c7dfd
    name ( RO): pool-operator
    description ( RO): The Pool Operator role manages host-  and pool-
        wide resources,
    including setting up storage, creating resource pools and managing
        patches, and
    high availability (HA).

    uuid( RO): 7955168d-7bec-10ed-105f-c6a7e6e63249
    name ( RO): vm-power-admin
    description ( RO): The VM Power Administrator role has full access
        to VM and
    template management and can choose where to start VMs and use the
        dynamic memory
    control and VM snapshot features

    uuid ( RO): aaa00ab5-7340-bfbc-0d1b-7cf342639a6e
    name ( RO): vm-admin
    description ( RO):  The VM Administrator role can manage VMs and
        templates

    uuid ( RO): fb8d4ff9-310c-a959-0613-54101535d3d5
    name ( RO): vm-operator
    description ( RO):  The VM Operator role can use VMs and interact
        with VM consoles

    uuid ( RO): 7233b8e3-eacb-d7da-2c95-f2e581cdbf4e
    name ( RO): read-only
    description ( RO): The Read-Only role can log in with basic read-
        only access
<!--NeedCopy-->
```

> **Note:**
>
> This list of roles is static. You cannot add, remove, or modify roles.

To display a list of current subjects

Run the following command:

```
xe subject-list
<!--NeedCopy-->
```

This command returns 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
<!--NeedCopy-->
```

To add a subject to RBAC

To enable existing AD users to use RBAC, create a subject instance within XenServer, either for the AD user directly, or for the containing groups:

Run the following command to add a new subject instance:

```
xe subject-add subject-name=AD user/group
<!--NeedCopy-->
```

To assign an RBAC role to a subject

After adding a subject, you can assign it to an RBAC role. You can refer to the role by either by its UUID or name:

Run the command:

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

Or

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

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
<!--NeedCopy-->
```

To change the RBAC role of a subject

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

Run the following 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
<!--NeedCopy-->
```

The user must log out and log back in to ensure that the new role takes effect. This requires the "Logout Active User Connections"permission available to a Pool Administrator or Pool Operator.

If you remove the Pool Admin role from a user, consider also changing the root password and rotating the pool secret. For more information, see Pool Security.

> **Warning:**
>
> When you add or remove a pool-admin subject, it can take a few seconds for all hosts in the pool to accept ssh sessions associated with this subject.

Auditing

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

- The message records the Subject ID and user name associated with the session that invoked the operation.
- If a subject invokes an operation that is not authorized, the operation is logged.
- Any successful operation is also recorded. If the operation failed then the error code is logged.

Audit log xe CLI commands

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

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

To obtain all audit records from the pool

Run the following command:

```
xe audit-log-get filename=/tmp/auditlog-pool-actions.out
<!--NeedCopy-->
```

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
<!--NeedCopy-->
```

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
<!--NeedCopy-->
```

Networking

April 25, 2024

This section 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:**
>
> vSwitch is the default network stack of XenServer. Follow the instructions in vSwitch networks to configure the Linux network stack.

If you are already familiar with XenServer networking concepts, you can skip ahead to Manage networking for information about the following sections:

- Create networks for standalone XenServer hosts
- Create networks for XenServer hosts that are configured in a resource pool
- Create VLANs for XenServer hosts, either standalone or part of a resource pool
- Create bonds for standalone XenServer hosts
- Create bonds for XenServer hosts that are configured in a resource pool

> **Note:**
>
> The term 'management interface' is used to indicate the IP-enabled NIC that carries the management traffic. The term 'secondary interface' is used to indicate an IP-enabled NIC configured for storage traffic.

Networking support

XenServer supports up to 16 physical network interfaces (or up to 4 bonded network interfaces) per 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). Do not edit the host networking configuration files directly.

vSwitch networks

vSwitch networks support open flow.

- Supports fine-grained security policies to control the flow of traffic sent to and from a VM.
- Provides detailed visibility about 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 when the VM migrates from one physical host in the resource pool to another.

To determine what networking stack is configured, run the following command:

```
xe host-list params=software-version
<!--NeedCopy-->
```

In the command output, look for `network_backend`. When the vSwitch is configured as the network stack, the output appears as follows:

```
network_backend: openvswitch
<!--NeedCopy-->
```

When the Linux bridge is configured as the network stack, the output appears as follows:

```
network_backend: bridge
<!--NeedCopy-->
```

To revert to the Linux network stack, run the following command:

```
xe-switch-network-backend bridge
<!--NeedCopy-->
```

Restart your host after running this command.

XenServer networking overview

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

XenServer creates a network for each physical NIC during installation. When you add a host to a pool, the default networks are merged. This is to ensure all physical NICs with the same device name are attached to the same network.

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

You can configure the following different types of networks in XenServer:

- **External networks** have an association with a physical network interface. External networks provide a bridge between a virtual machine and the physical network interface connected to the network. External networks enable a virtual machine to connect to resources available through the host's physical NIC.
- **Bonded networks** create a bond between two or more NICs to create a single, high-performing channel between the virtual machine and the network.
- **Single-Server Private networks** have no association to a physical network interface. Single-server private networks can be used to provide connectivity between the virtual machines on a given host, with no connection to the outside world.

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

Network objects

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

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

XenCenter and the xe CLI allow you to configure networking options. You can control the NIC used for management operations, and create advanced networking features such as 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*. Internal networks 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*. External networks provide a bridge between VIFs and the PIF connected to the network, enabling connectivity to resources available through the PIF's NIC.

VLANs

VLANs, as defined by the IEEE 802.1Q standard, allow a single physical network to support multiple logical networks. XenServer hosts support VLANs in multiple ways.

> **Note:**
>
> - We recommend not to use a GFS2 SR with a VLAN due to a known issue where you cannot add or remove hosts on a clustered pool if the cluster network is on a non-management VLAN.
> - All supported VLAN configurations are equally applicable to pools and standalone hosts, and bonded and non-bonded configurations.

Using VLANs with virtual machines Switch ports configured as 802.1Q VLAN trunk ports can be used 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. You can connect XenServer networks to the PIF representing the physical NIC to see all traffic on the NIC. Alternatively, connect networks to a PIF representing a VLAN to see only the traffic with the specified VLAN tag. You can also connect a network such that it only sees the native VLAN traffic, by attaching it to VLAN 0.

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

If you want the guest to perform the VLAN tagging and untagging functions, the guest must be aware of the VLANs. When configuring the network for your VMs, configure the switch ports as VLAN trunk ports, but do not create VLANs for the XenServer host. Instead, use VIFs on a normal, non-VLAN network.

Using VLANs with management interfaces Management interface can be configured on a VLAN using a switch port configured as trunk port or access mode port. Use XenCenter or xe CLI to set up a VLAN and make it the management interface. For more information, see Management interface.

Using VLANs with dedicated storage NICs Dedicated storage NICs can be configured to use native VLAN or access mode ports as described in the previous section for management interfaces. Dedicated storage NICs are also known as IP-enabled NICs or secondary interfaces. You can configure dedicated storage NICs to use trunk ports and XenServer VLANs as described in the previous section for virtual machines. For more information, see 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.

Jumbo frames

Jumbo frames can be used to optimize the performance of traffic on storage networks and VM networks. Jumbo frames are Ethernet frames containing more than 1,500 bytes of payload. Jumbo frames are typically used to achieve better throughput, reduce the load on system bus memory, and reduce the CPU overhead.

> **Note:**
>
> XenServer supports jumbo frames only when using vSwitch as the network stack on all hosts in the pool.

Requirements for using jumbo frames Note the following when using jumbo frames:

- Jumbo frames are configured at a pool level
- vSwitch must be configured as the network back-end on all hosts in the pool
- Every device on the subnet must be configured to use jumbo frames
- Enabling jumbo frames on the management network is not supported

To use jumbo frames, set the Maximum Transmission Unit (MTU) to a value between 1500 and 9216. You can use XenCenter or the xe CLI to set the MTU.

NIC Bonds

NIC bonds, sometimes also known as NIC teaming, improve XenServer host resiliency and bandwidth by enabling administrators to configure two or more NICs together. NIC bonds logically function as one network card and all bonded NICs share a MAC address.

If one NIC in the bond fails, the host's network traffic is automatically redirected through the second NIC. XenServer supports up to eight bonded networks.

XenServer supports active-active, active-passive, and LACP bonding modes. The number of NICs supported and the bonding mode supported varies according to network stack:

- LACP bonding is only available for the vSwitch whereas active-active and active-passive are available for both the vSwitch and the Linux bridge.
- When the vSwitch is the network stack, you can bond either two, three, or four NICs.
- When the Linux bridge is the network stack, you can only bond two NICs.

All bonding modes support failover. However, not all modes allow all links to be active for all traffic types. XenServer supports bonding the following types of NICs together:

- **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.
- **Management interfaces**. You can bond a management interface to another NIC so that the second NIC provides failover for management traffic. Although configuring an LACP link aggregation bond provides load balancing for management traffic, active-active NIC bonding does not. You can create a VLAN on bonded NICs and the host management interface can be assigned to that VLAN.
- **Secondary interfaces**. You can bond NICs that you have configured as secondary interfaces (for example, for storage). However, for most iSCSI software initiator storage, we recommend configuring multipathing instead of NIC bonding as described in the Designing XenServer Network Configurations.

  Throughout this section, the term IP-based storage traffic is used to describe iSCSI and NFS traffic collectively.

You can create a bond if a VIF is already using one of the interfaces that will be bonded: the VM traffic migrates automatically to the new bonded interface.

In XenServer, An additional PIF represents a NIC bond. XenServer NIC bonds completely subsume the underlying physical devices (PIFs).

> **Notes:**
>
> - Creating a bond that contains only one NIC is not supported.
> - The bonded NICs can be different models to each other.
> - NIC bonds are not supported on NICs that carry FCoE traffic.

Best practices

Follow these best practices when setting up your NIC bonds:

- Connect the links of the bond to different physical network switches, not just ports on the same switch.
- Ensure that the separate switches draw power from different, independent power distribution units (PDUs).
- If possible, in your data center, place the PDUs on different phases of the power feed or even feeds provided by different utility companies.
- Consider using uninterruptible power supply units to ensure that the network switches and hosts can continue to function or perform an orderly shutdown in the event of a power failure.

These measures add resiliency against software, hardware, or power failures that can affect your network switches.

Key points about IP addressing

Bonded NICs either have one IP address or no IP addresses, as follows:

- **Management and storage networks**.
  - If you bond a management interface or secondary 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 bonds are used for non-VM traffic, for example, to connect to shared network storage or XenCenter for management, configure an IP address for the bond. However, if you have already assigned an IP address to one of the NICs (that is, created a management interface or secondary interface), that IP address is assigned to the entire bond automatically.
  - If you bond a management interface or secondary interface to a NIC without an IP address, the bond assumes the IP address of the respective interface.
  - If you bond a tagged VLAN management interface and a secondary interface, the management VLAN is created on that bonded NIC.
- **VM networks**. When bonded NICs are used for VM 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. IP addresses for virtual machines are associated with VIFs.

Bonding types

XenServer provides three different types of bonds, all of which can be configured using either the CLI or XenCenter:

- Active-Active mode, with VM traffic balanced between the bonded NICs. See Active-active bonding.
- Active-Passive mode, where only one NIC actively carries traffic. See Active-passive bonding.
- LACP Link Aggregation, in which active and stand-by NICs are negotiated between the switch and the host. See LACP Link Aggregation Control Protocol bonding.

> **Note:**
>
> Bonding is set up with an Up Delay of 31,000 ms and a Down Delay of 200 ms. The seemingly long Up Delay is deliberate because of the time some switches take to enable the port. Without a delay, when a link comes back after failing, the bond can rebalance traffic onto it before the switch is ready to pass traffic. To move both connections to a different switch, move one, then wait 31 seconds for it to be used again before moving the other. For information about changing the delay, see Changing the up delay for bonds.

Bond status

XenServer provides status for bonds in the event logs for each host. If one or more links in a bond fails or is restored, it is noted in the event log. Likewise, you can query the status of a bond's links by using the `links-up` parameter as shown in the following example:

```
xe bond-param-get uuid=bond_uuid param-name=links-up
<!--NeedCopy-->
```

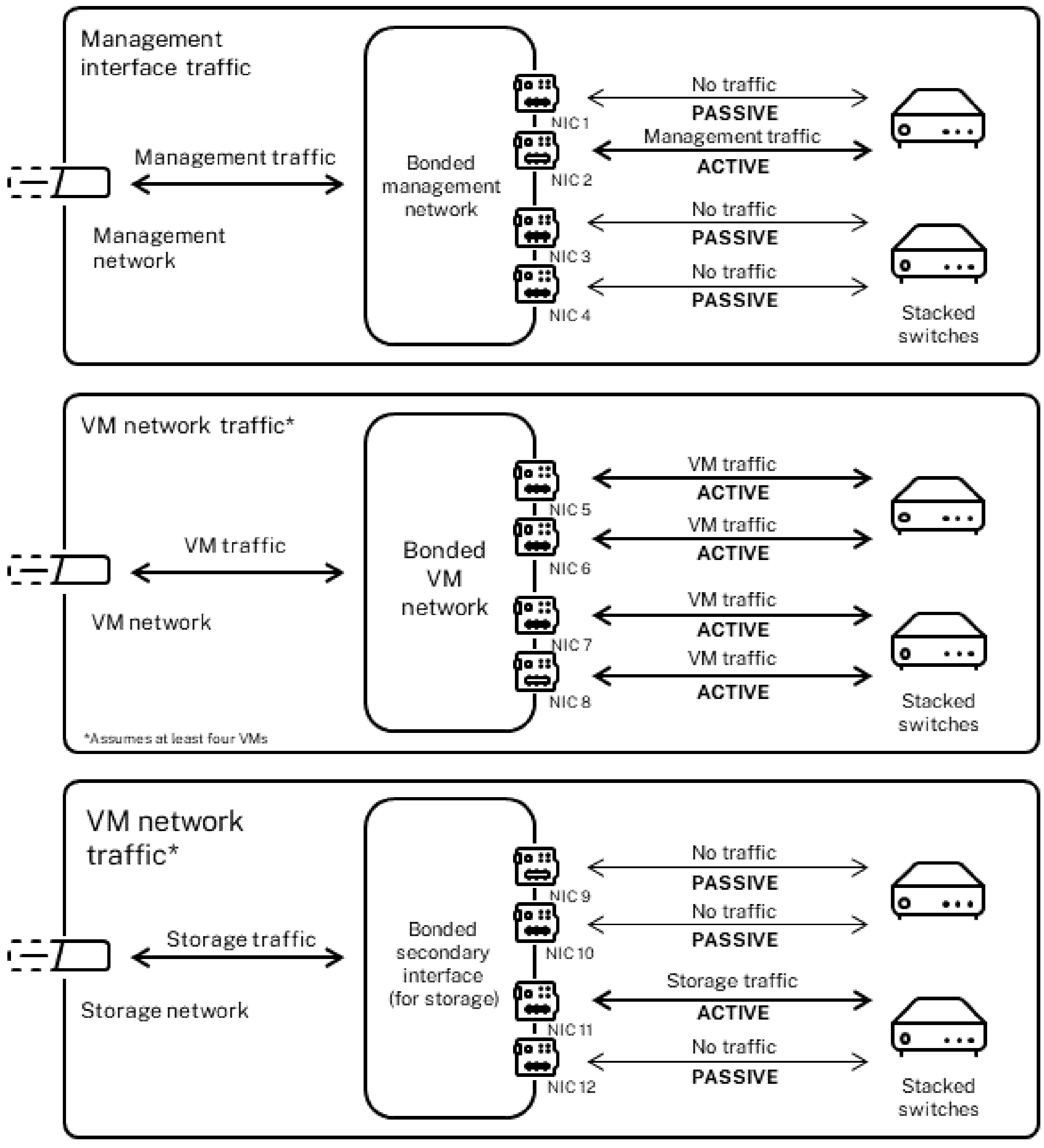
XenServer checks the status of links in bonds approximately every five seconds. Therefore, if more links in the bond fail in the five-second window, the failure is not logged until the next status check.

Bonding event logs appear in the XenCenter Logs tab. For users not running XenCenter, event logs also appear in `/var/log/xensource.log` on each host.

Active-active bonding

Active-active 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 failover support. Active-active mode is the default bonding mode when either the Linux bridge or vSwitch network stack is enabled.

When active-active bonding is used with the Linux bridge, you can only bond two NICs. When using the vSwitch as the network stack, you can bond either two, three, or four NICs in active-active mode. However, in active-active mode, bonding three, or four NICs is only beneficial for VM traffic, as shown in the illustration that follows.



Active-active bonds (vSwitch network stack)

XenServer can only send traffic over two or more NICs when there is more than one MAC address associated with the bond. XenServer can use the virtual MAC addresses in the VIF to send traffic across multiple links. Specifically:

- **VM traffic**. Provided you enable bonding on NICs carrying only VM (guest) traffic, all links are active and NIC bonding can balance spread VM traffic across NICs. An individual VIF's traffic is never split between NICs.

- **Management or storage traffic**. Only one of the links (NICs) in the bond is active and the other NICs remain unused unless traffic fails over to them. Configuring a management interface or secondary interface on a bonded network provides resilience.
- **Mixed traffic**. If the bonded NIC carries a mixture of IP-based storage traffic and guest traffic, only the guest and control domain traffic are load balanced. The control domain is essentially a virtual machine so it uses a NIC like the other guests. XenServer balances the control domain's traffic the same way as it balances VM traffic.

Traffic balancing XenServer balances the traffic between NICs by using the source MAC address of the packet. Because for management traffic, only one source MAC address is present, active-active mode can only use one NIC, and traffic is not balanced. Traffic balancing is based on two factors:

- The virtual machine and its associated VIF sending or receiving the traffic
- The quantity of data (in kilobytes) being sent.

XenServer evaluates the quantity of data (in kilobytes) each NIC is sending and receiving. If the quantity of data sent across one NIC exceeds the quantity of data sent across the other NIC, XenServer rebalances which VIFs use which NICs. The VIF's entire load is transferred. One VIF's load is never split between two NICs.

Though active-active 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 periodically rebalances traffic, VIFs are not permanently assigned to a specific NIC in the bond.

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

When rebalancing, the number of bytes going over each secondary (interface) is tracked over a given period. If a packet to be sent contains a new source MAC address, it is assigned to the secondary interface with the lowest utilization. Traffic is rebalanced at regular intervals.

Each MAC address has a corresponding load and XenServer can shift entire loads between NICs depending on the quantity of data a VM sends and receives. For active-active traffic, all the traffic from one VM can be sent on only one NIC.

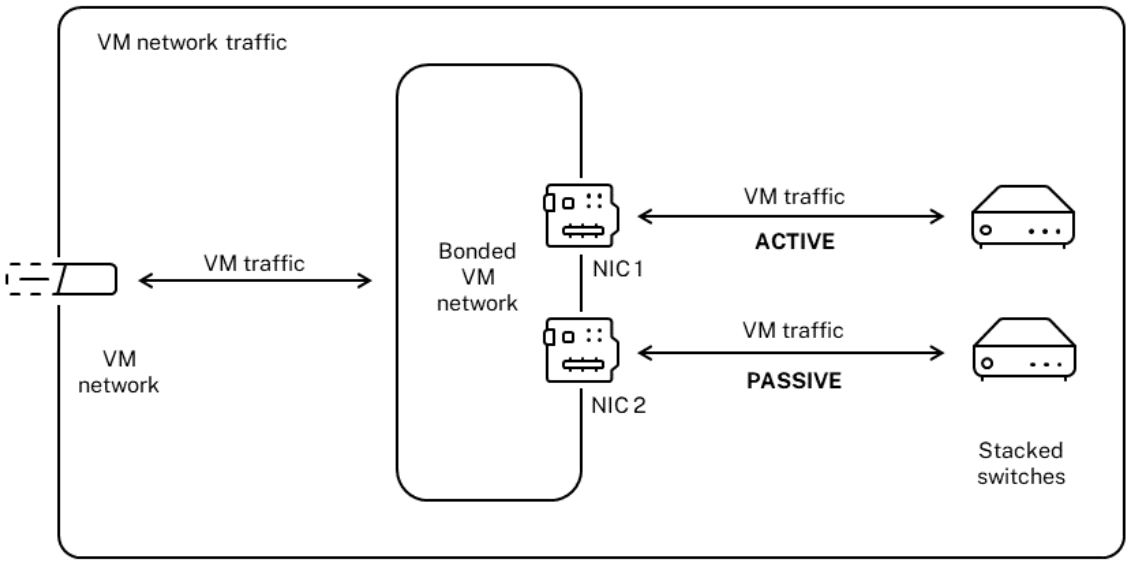
> **Note:**
>
> Active-active bonding does not require switch support for EtherChannel or 802.3ad (LACP).

Active-passive bonding

An active-passive bond routes traffic over only one of the NICs. If the active NIC loses network connectivity, traffic fails over to the other NIC in the bond. Active-passive bonds route traffic over the active NIC. The traffic shifts to the passive NIC if the active NIC fails.

Active-passive bonding is available in the Linux bridge and the vSwitch network stack. When used with the Linux bridge, you can bond two NICs together. When used with the vSwitch, you can only bond two, three, or four NICs together. However, regardless of the traffic type, when you bond NICs in active-passive mode, only one link is active and there is no load balancing between links.

The illustration that follows shows two bonded NICs configured in active-passive mode.



Active-passive bond

Active-active mode is the default bonding configuration in XenServer. If you are configuring bonds using the CLI, you must specify a parameter for the active-passive mode. Otherwise, an active-active bond is created. You do not need to configure active-passive mode because a network is carrying management traffic or storage traffic.

Active-passive can be a good choice for resiliency as it offers several benefits. With active-passive bonds, traffic does not move around between NICs. Similarly, active-passive bonding lets you configure two switches for redundancy but does not require stacking. If the management switch dies, stacked switches can be a single point of failure.

Active-passive mode does not require switch support for EtherChannel or 802.3ad (LACP).

Consider configuring active-passive mode in situations when you do not need load balancing or when you only intend to send traffic on one NIC.

> **Important:**
>
> After you have created VIFs or your pool is in production, be careful about changing bonds or creating bonds.

LACP Link Aggregation Control Protocol bonding

LACP Link Aggregation Control Protocol is a type of bonding that bundles a group of ports together and treats it like a single logical channel. LACP bonding provides failover and can increase the total amount of bandwidth available.

Unlike other bonding modes, LACP bonding requires configuring both sides of the links: creating a bond on the host, and creating a Link Aggregation Group (LAG) for each bond on the switch. See Switch configuration for LACP bonds. You must configure the vSwitch as the network stack to use LACP bonding. Also, your switches must support the IEEE 802.3ad standard.

A comparison of active-active SLB bonding and LACP bonding:

Active-active SLB bonding **Benefits:**

- Can be used with any switch on the Hardware Compatibility List.
- Does not require switches that support stacking.
- Supports four NICs.

Considerations:

- Optimal load balancing requires at least one NIC per VIF.
- Storage or management traffic cannot be split on multiple NICs.
- Load balancing occurs only if multiple MAC addresses are present.

LACP bonding **Benefits:**

- All links can be active regardless of traffic type.
- Traffic balancing does not depend on source MAC addresses, so all traffic types can be balanced.

Considerations:

- Switches must support the IEEE 802.3ad standard.
- Requires switch-side configuration.
- Supported only for the vSwitch.
- Requires a single switch or stacked switch.

Traffic balancing XenServer supports two LACP bonding hashing types. The term hashing describes how the NICs and the switch distribute the traffic—(1) load balancing based on IP and port of source and destination addresses and (2) load balancing based on source MAC address.

Depending on the hashing type and traffic pattern, LACP bonding can potentially distribute traffic more evenly than active-active NIC bonding.

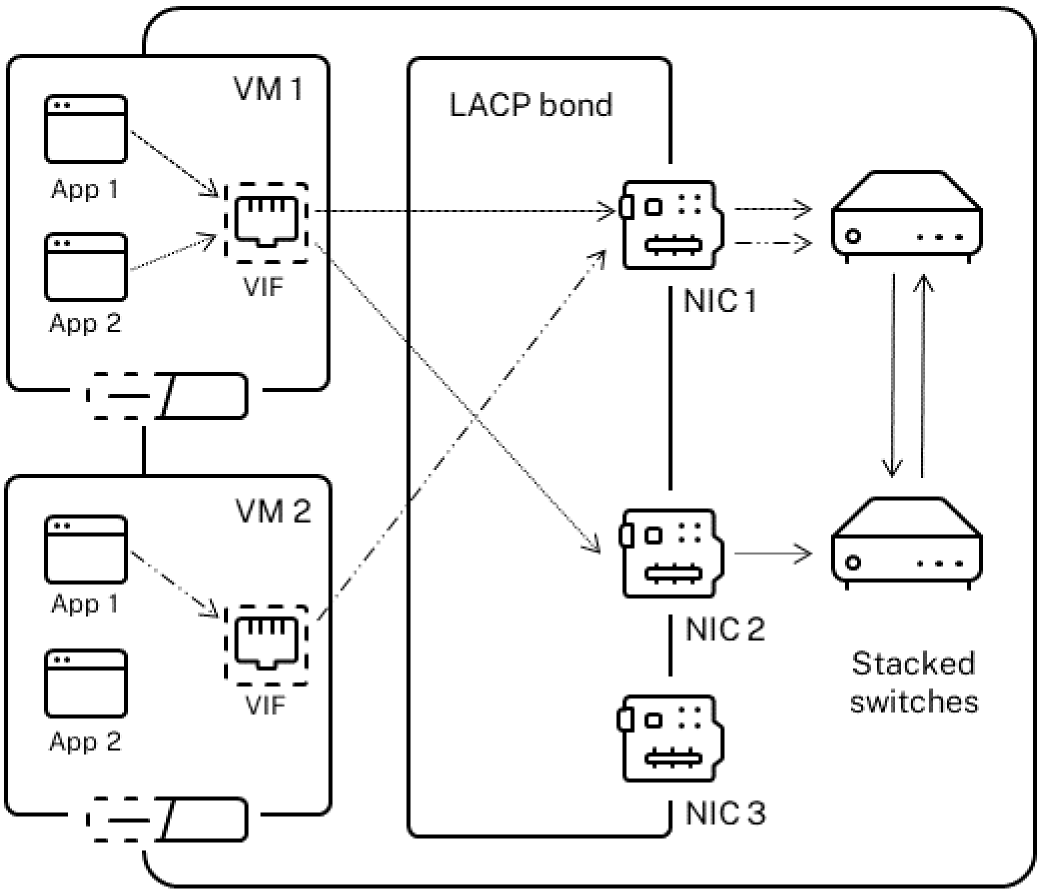
> **Note:**
>
> You configure settings for outgoing and incoming traffic separately on the host and the switch: the configuration does not have to match on both sides.

Load balancing based on IP and port of source and destination addresses.

This hashing type is the default LACP bonding hashing algorithm. If there is a variation in the source or destination IP or port numbers, traffic from one guest can be distributed over two links.

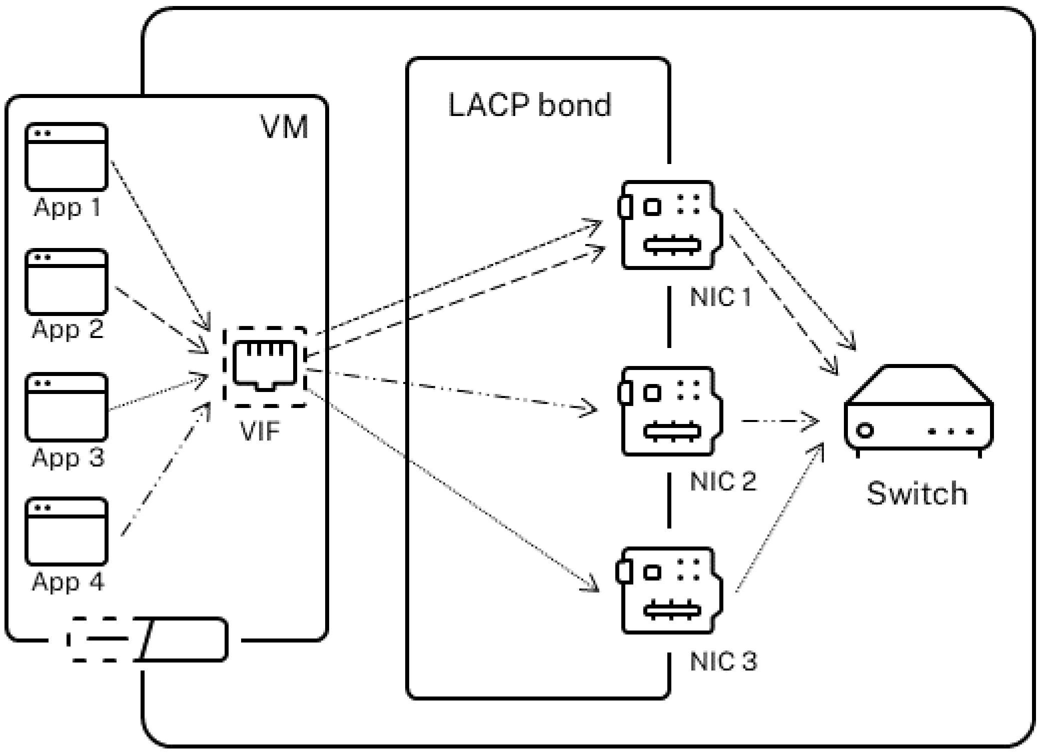
If a virtual machine is running several applications which use different IP or port numbers, this hashing type distributes traffic over several links. Distributing the traffic gives the guest the possibility of using the aggregate throughput. This hashing type lets one guest use the whole throughput of multiple NICs.

As shown in the illustration that follows, this hashing type can distribute the traffic of two different applications on a virtual machine to two different NICs.



Load balancing based on IP and port of source and destination addresses

Configuring LACP bonding based on IP and port of source and destination address is beneficial when you want to balance the traffic of two different applications on the same VM. For example, when only one virtual machine is configured to use a bond of three NICs.

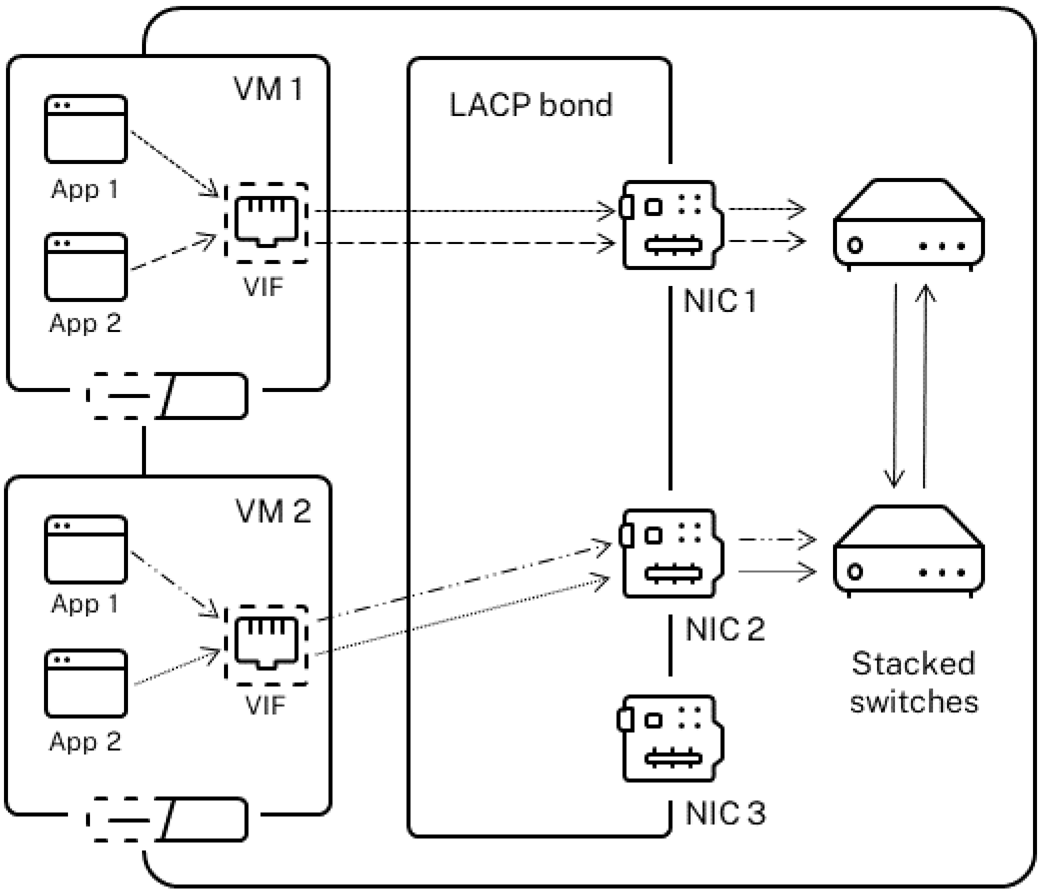


Load balancing based on IP and port of source and destination addresses

The balancing algorithm for this hashing type uses five factors to spread traffic across the NICs: the source IP address, source port number, destination IP address, destination port number, and source MAC address.

Load balancing based on source MAC address.

This type of load balancing works well when there are multiple virtual machines on the same host. Traffic is balanced based on the virtual MAC address of the VM from which the traffic originated. XenServer sends outgoing traffic using the same algorithm as it does in active-active bonding. Traffic coming from the same guest is not split over multiple NICs. As a result, this hashing type is not suitable if there are fewer VIFs than NICs: load balancing is not optimal because the traffic cannot be split across NICs.



Load balancing based on source MAC address

Switch configuration

Depending on your redundancy requirements, you can connect the NICs in the bond to either the same or separate stacked 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 prevents a single point-of-failure in your configuration in the following ways:

- When you connect one of the links in a bonded management interface to a second switch, if the switch fails, the management network remains online and the hosts can still communicate with each other.
- If you connect a link (for any traffic type) to a second switch and the NIC or switch fails, the virtual machines remain on the network as their traffic fails over to the other NIC/switch.

Use stacked switches when you want to connect bonded NICs to multiple switches and you configured the LACP bonding mode. The term 'stacked switches'is used to describe configuring multiple physical switches to function as a single logical switch. You must join the switches together physically and through the switch-management software so the switches function as a single logical switching unit, as per the switch manufacturer's guidelines. Typically, switch stacking is only available through proprietary extensions and switch vendors may market this functionality under different terms.

> **Note:**
>
> If you experience issues with active-active bonds, the use of stacked switches may be necessary. Active-passive bonds do not require stacked switches.

Switch configuration for LACP bonds Because the specific details of switch configuration vary by manufacturer, there are a few key points to remember when configuring switches for use with LACP bonds:

- The switch must support LACP and the IEEE 802.3ad standard.
- When you create the LAG group on the switch, you must create one LAG group for each LACP bond on the host. For example, if you have a five-host pool and you created an LACP bond on NICs 4 and 5 on each host, you must create five LAG groups on the switch. One group for each set of ports corresponding with the NICs on the host.

  You may also need to add your VLAN ID to your LAG group.
- XenServer LACP bonds require setting the Static Mode setting in the LAG group to be set to Disabled.

As previously mentioned in *Switch configuration*, stacking switches are required to connect LACP bonds to multiple switches.

Initial networking configuration after setup

The XenServer host networking configuration is specified during initial host installation. Options such as IP address configuration (DHCP/static), the NIC used as the management interface, and host name 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 management interface is configured with the IP addressing options specified during installation
- A network is created for each PIF ("network 0", "network 1", and so on)
- Each network is connected to one PIF

- The IP addressing options are left unconfigured for all PIFs other than the PIF used as the management interface

When a host has a single NIC, the following configuration is present after installation:

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

When an installation of XenServer is done on a tagged VLAN network, the following configuration is present after installation:

- PIFs are created for each NIC in the host
- The PIF for the tagged VLAN on the NIC selected for use as the management interface is configured with the IP address configuration specified during installation
- A network is created for each PIF (for example: network 1, network 2, and so on). Additional VLAN network is created (for example, for Pool-wide network associated with eth0 on VLAN<TAG>)
- Each network is connected to one PIF. The VLAN PIF is set for use in host management operations

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 through the IP address of the management interface. The configuration also provides external networking for VMs created on the host.

The PIF used for management operations is the only PIF ever configured with an IP address 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.

Changing networking configuration

You can change your networking configuration by modifying the network object. To do so, you run a command that affects either the network object or the VIF.

Modifying the network object

You can change aspects of a network, such as the frame size (MTU), name-label, name-description, purpose, and other values. Use the xe `network-param-set` command and its associated parameters to change the values.

When you run the xe `network-param-set` command, the only required parameter is `uuid`.

Optional parameters include:

- `default_locking_mode`. See Simplifying VIF locking mode configuration in the Cloud.
- `name-label`
- `name-description`
- `MTU`
- `purpose`. See Adding a purpose to a network.
- `other-config`

If a value for a parameter is not given, the parameter is set to a null value. To set a (key, value) pair in a map parameter, use the syntax `map-param:key=value`.

Changing the up delay for bonds

Bonding is set up with an Up Delay of 31,000 ms by default to prevent traffic from being rebalanced onto a NIC after it fails. While seemingly long, the up delay is important for all bonding modes and not just active-active.

However, if you understand the appropriate settings to select for your environment, you can change the up delay for bonds by using the procedure that follows.

Set the up delay in milliseconds:

```
xe pif-param-set uuid=<uuid of bond interface PIF> other-config:bond-
    updelay=<delay in ms>
<!--NeedCopy-->
```

To make the change take effect, you must unplug and then replug the physical interface:

```
xe pif-unplug uuid=<uuid of bond interface PIF>
<!--NeedCopy-->
```

```
xe pif-plug uuid=<uuid of bond interface PIF>
<!--NeedCopy-->
```

Manage networking

February 19, 2024

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

Create networks in a standalone host

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

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

For information about how to add or delete networks using XenCenter, see Add a New Network in the XenCenter documentation.

Open the XenServer host text console.

Create the network by using the network-create command, which returns the UUID of the newly created network:

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

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

Create networks in resource pools

All XenServer hosts in a resource pool must have the same number of physical NICs. This requirement is not strictly enforced when a host is joined to a pool. One of the NICs is always designated as the *management interface*, used for XenServer management traffic.

As all hosts in a pool share a common set of network. It is important to have the same physical networking configuration for XenServer hosts in a pool. PIFs on the individual hosts are connected to pool-wide networks based on device name. For example, all XenServer hosts in a pool with eth0 NIC have a corresponding PIF plugged to the pool-wide `Network 0` network. The same is true for hosts with eth1 NICs and `Network 1`, and 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. The complications can arise because not all pool networks are valid for all pool hosts. For example, if hosts *host1* and *host2* are in the same pool and *host1* has four NICs and *host2* only has two, only the networks connected to PIFs corresponding to eth0 and eth1 are valid on *host2*. VMs on *host1* with VIFs connected to networks corresponding to eth2 and eth3 cannot migrate to host *host2*.

Create VLANs

For hosts in a resource pool, you can use the `pool-vlan-create` command. This command creates the VLAN and automatically creates and plug-ins the required PIFs on the hosts in the pool. For more information, see `pool-vlan-create`.

Open the XenServer host console.

Create a network for use with the VLAN. The UUID of the new network is returned:

```
xe network-create name-label=network5
<!--NeedCopy-->
```

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
<!--NeedCopy-->
```

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 is created and plugged to 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
<!--NeedCopy-->
```

Attach VM VIFs to the new network. For more information, see Creating networks in a standalone host.

Create NIC bonds on a standalone host

We recommend using XenCenter to create NIC bonds. For more information, see Configuring NICs.

This section describes how to use the xe CLI to bond NIC interfaces on XenServer hosts that are not in a pool. For information on using the xe CLI to create NIC bonds on XenServer hosts that comprise a resource pool, see *Creating NIC bonds in resource pools*.

Create a NIC bond

When you bond a NIC, the bond absorbs the PIF/NIC in use as the management interface. The management interface is automatically moved to the bond PIF.

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

   ```
   xe network-create name-label=bond0
   <!--NeedCopy-->
   ```

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

   ```
   xe pif-list
   <!--NeedCopy-->
   ```

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,pif_uuid_3,pif_uuid_4
     <!--NeedCopy-->
     ```

     Type two UUIDs when you are bonding two NICs and four UUIDs when you are bonding four NICs. The UUID for the bond is returned after running the command.

   - To configure the bond in active-passive or LACP bond mode, use the same syntax, add the optional `mode` parameter, and specify `lacp` or `active-backup`:

     ```
     xe bond-create network-uuid=network_uuid pif-uuids=pif_uuid_1
         , /
          pif_uuid_2,pif_uuid_3,pif_uuid_4 /
          mode=balance-slb | active-backup | lacp
     <!--NeedCopy-->
     ```

Control the MAC address of the bond

When you bond the management interface, it subsumes the PIF/NIC in use as the management interface. If the host uses DHCP, the bond's MAC address is the same as the PIF/NIC in use. The 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) management-interface NIC. However, as the bond is enabled and the MAC/IP address in use changes, existing network sessions to the host are 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, XenServer uses the MAC address of the management interface if it is one of the interfaces in the bond. If the management interface is not part of the bond, but another management interface is, the bond uses the MAC address (and also the IP address) of that management interface. If none of the NICs in the bond is a management interface, the bond uses the MAC of the first named NIC.

Revert NIC bonds

When reverting the XenServer host to a non-bonded configuration, the `bond-destroy` command automatically configures the primary NIC as the interface for the management interface. Therefore, all VIFs are moved to the management interface. If management interface of a host is on tagged VLAN bonded interface, on performing `bond-destroy`, management VLAN is moved to primary NIC.

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

1. The management interface NIC (if the management interface is one of the bonded NICs).
2. Any other NIC with an IP address (if the 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
   <!--NeedCopy-->
   ```

Create NIC bonds in resource pools

Whenever possible, create NIC bonds as part of initial resource pool creation, before joining more 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 pool coordinator and then each member of the pool.
- Using the CLI to configure bonds on the pool coordinator and then restarting each pool member so that it inherits its settings from the pool coordinator.
- Using XenCenter to configure the bonds on the pool coordinator. XenCenter automatically synchronizes the networking settings on the member hosts with the pool coordinator, so you do not need to restart the member hosts.

For simplicity and to prevent misconfiguration, we recommend using XenCenter to create NIC bonds. For more information, see Configuring NICs.

This section describes using the xe CLI to create bonded NIC interfaces on XenServer hosts that comprise a resource pool. For information on using the xe CLI to create NIC bonds on a standalone host, see *Creating NIC bonds on a standalone host*.

> **Warning:**
>
> Do not attempt to create network bonds when high availability is enabled. The process of bond creation disturbs the in-progress high availability heartbeat and causes hosts to self-fence (shut themselves down). The hosts can fail to restart properly and may need the `host-emergency-ha-disable` command to recover.

Select the host you want to be the pool coordinator. The pool coordinator 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
<!--NeedCopy-->
```

Create the NIC bond as described in Create a NIC bond.

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
<!--NeedCopy-->
```

The network and bond information is automatically replicated to the new host. The management interface is automatically moved from the host NIC where it was originally configured to the bonded PIF. That is, the management interface is now absorbed into the bond so that the entire bond functions as the management interface.

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

```
xe host-list
<!--NeedCopy-->
```

> **Warning:**
>
> Do not attempt to create network bonds while high availability is enabled. The process of bond creation disturbs the in-progress high availability heartbeat and causes hosts to self-fence (shut themselves down). The hosts can fail to restart properly and you may need to run the `host-emergency-ha-disable` command to recover.

Configure a dedicated storage NIC

You can use 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 secondary interface. (The IP-enabled NIC XenServer used for management is known as the management interface.)

When you want to dedicate a secondary interface for a specific purpose, ensure that the appropriate network configuration is in place. This is to ensure that the NIC is used only for the desired traffic. To dedicate a NIC to storage traffic, configure the NIC, storage target, switch, and VLAN such that the target is only accessible over the assigned NIC. If your physical and IP configuration does not limit the traffic sent across the storage NIC, you can send traffic, such as management traffic across the secondary interface.

When you create a new secondary interface for storage traffic, you must assign it an IP address that is:

- On the same subnet as the storage controller, if applicable, and
- Not on the same subnet as any other secondary interfaces or the management interface.

When you are configuring secondary interfaces, each secondary interface must be on a separate subnet. For example, if you want to configure two more secondary interfaces for storage, you require IP addresses on three different subnets –one subnet for the management interface, one subnet for Secondary Interface 1, and one subnet for Secondary Interface 2.

If you are using bonding for resiliency for your storage traffic, you may want to consider using LACP instead of the Linux bridge bonding. To use LACP bonding, you must configure the vSwitch as your networking stack. For more information, see vSwitch networks.

> **Note:**
>
> When selecting a NIC to configure as a secondary interface for use with iSCSI or NFS SRs, ensure that the dedicated NIC uses a separate IP subnet that is not routable from the management interface. If this is not enforced, then storage traffic may be directed over the main management interface after a host restart, because of the order in which network interfaces are initialized.

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

Set up an IP configuration for the PIF, adding appropriate values for the mode parameter. If using static IP addressing, add the IP, netmask, gateway, and DNS parameters:

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

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

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

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

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

- After a host restart, ensure that the secondary interface is correctly configured. Use the `xe pbd-unplug` and `xe pbd-plug` commands to reinitialize the storage connections on the host. This command restarts the storage connection and routes it over the correct interface.
- Alternatively, you can use `xe pif-forget` to delete the interface from the XenServer database and manually configure it in the control domain. `xe pif-forget` is an advanced option and requires you to be familiar with how to configure Linux networking manually.

Use SR-IOV enabled NICs

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

Assigning one or more NIC VFs to a VM allows its network traffic to bypass the virtual switch. When configured, each VM behaves as though it is using the NIC directly, reducing processing overhead, and improving performance.

Benefits of SR-IOV

An SR-IOV VF has a better performance than VIF. It can ensure the hardware-based segregation between traffic from different VMs through the same NIC (bypassing the XenServer network stack).

Using this feature, you can:

- Enable SR-IOV on NICs that support SR-IOV.
- Disable SR-IOV on NICs that support SR-IOV.
- Manage SR-IOV VFs as a VF resource pool.
- Assign SR-IOV VFs to a VM.
- Configure SR-IOV VFs (For example, MAC address, VLAN, rate).
- Run tests to confirm if SR-IOV is supported as part of the Automated Certification Kit.

System configuration

Configure the hardware platform correctly to support SR-IOV. The following technologies are required:

- I/O MMU virtualization (AMD-Vi and Intel VT-d)
- Alternative Routing-ID Interpretation (ARI)
- Address Translation Services (ATS)
- Access Control Services (ACS)

Check the documentation that comes with your system for information on how to configure the system firmware to enable the mentioned technologies.

Enable an SR-IOV network on a NIC

In XenCenter, use the **New Network** wizard in the **Networking** tab to create and enable an SR-IOV network on a NIC.

Assign an SR-IOV network to the virtual interface (VM level)

In XenCenter, at the VM level, use the **Add Virtual Interface** wizard in the **Networking** tab to add an SR-IOV enabled network as a virtual interface for that VM. For more information, see Add a New Network.

Supported NICs and guests

For a list of supported hardware platforms and NICs, see Hardware Compatibility List. See the documentation provided by the vendor for a particular guest to determine whether it supports SR-IOV.

Limitations

- For certain NICs using legacy drivers (for example, Intel I350 family) the host must be rebooted to enable or disable SR-IOV on these devices.
- A pool level SR-IOV network having different types of NICs are not supported.
- An SR-IOV VF and a normal VIF from the same NIC may not be able to communicate with each other because of the NIC hardware limitations. To enable these VMs to communicate, ensure that communication uses the pattern VF to VF or VIF to VIF, and not VF to VIF.

- Quality of Service settings for some SR-IOV VFs do not take effect because they do not support network speed rate limiting.
- Performing live migration, suspend, and checkpoint is not supported on VMs using an SR-IOV VF.
- SR-IOV VFs do not support hot-plugging.
- SR-IOV VFs do not support network boot.
- For some NICs with legacy NIC drivers, rebooting may be required even after host restart which indicates that the NIC is not able to enable SR-IOV.
- If your VM has an SR-IOV VF, functions that require Live Migration are not possible. This is because the VM is directly tied to the physical SR-IOV enabled NIC VF.
- SR-IOV can be used in an environment that makes use of high availability. However, SR-IOV is not considered in the capacity planning. VMs that have SR-IOV VFs assigned are restarted on a best-effort basis when there is a host in the pool that has appropriate resources. These resources include SR-IOV enabled on the right network and a free VF.
- SR-IOV VFs are not supported with the PVS-Accelerator.

Configure SR-IOV VFs for legacy drivers

Usually the maximum number of VFs that a NIC can support can be determined automatically. For NICs using legacy drivers (for example, Intel I350 family), the limit is defined within the driver module configuration file. The limit may need to be adjusted manually. To set it to the maximum, open the file using an editor and change the line starting:

```
## VFs-maxvfs-by-user:
<!--NeedCopy-->
```

For example, to set the maximum VFs to 4 for the `igb` driver edit `/etc/modprobe.d/igb.conf` to read:

```
## VFs-param: max_vfs
## VFs-maxvfs-by-default: 7
## VFs-maxvfs-by-user: 4
options igb max_vfs=0
<!--NeedCopy-->
```

> **Notes:**
>
> - The value must be less than or equal to the value in the line `VFs-maxvfs-by-default`.
> - Do not change any other line in these files.

> - Make the changes before enabling SR-IOV.

CLI

See SR-IOV commands for CLI instructions on creating, deleting, displaying SR-IOV networks and assigning an SR-IOV VF to a VM.

Control the rate of outgoing data (QoS)

To limit the amount of *outgoing* data a VM can send per second, 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 *kilobytes* per second.

The Quality of Service value limits the rate of transmission *from* the VM. The Quality of Service setting does not limit the amount of data the VM can receive. If such a limit is desired, we recommend 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 value on VM virtual interfaces (VIFs) in one of two places. You can set this value either by using the xe CLI or in XenCenter.

- **XenCenter** You can set the Quality of Service transmit rate limit value in the properties dialog for the virtual interface.
- **xe commands** You can set the Quality of Service transmit rate using the CLI using the commands in the section that follow.

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
<!--NeedCopy-->
```

> **Note:**
>
> The `kbps` parameter denotes *kilobytes* per second (kBps), not kilobits per second (kbps).

Change networking configuration options

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

- Changing the hostname (that is, the Domain Name System (DNS) name)
- Adding or deleting DNS servers
- Changing IP addresses
- Changing which NIC is used as the management interface
- Adding a new physical NIC to the server
- Adding a purpose to a network
- Enabling ARP filtering (switch-port locking)

Hostname

The system hostname, also known as the domain or DNS name, is defined in the pool-wide database and changed using the `xe host-set-hostname-live` CLI command as follows:

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

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

DNS servers

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

```
xe pif-reconfigure-ip uuid=pif_uuid mode=static DNS=new_dns_ip IP=IP
    netmask=netmask
<!--NeedCopy-->
```

Change IP address configuration for a standalone host

You can use the xe CLI to change the network interface configuration. Do not change the underlying network configuration scripts directly.

To change the IP address configuration of a PIF, use the `pif-reconfigure-ip` CLI command. See `pif-reconfigure-ip` for details on the parameters of the `pif-reconfigure-ip` command. See the following section for information on changing host IP addresses in resource pools.

Change IP address configuration in resource pools

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

> **Note:**
>
> You must be careful when changing the IP address of a host, and other networking parameters. Depending upon the network topology and the change being made, connections to network storage can be lost. When this happens, the storage must be replugged using the **Repair Storage** function in XenCenter, or by using the `pbd-plug` CLI command. For this reason, we recommend that you migrate VMs away from the host before changing its IP configuration.

Use the `pif-reconfigure-ip` CLI command to set the IP address as desired. See `pif-reconfigure-ip` for details on the parameters of the `pif-reconfigure-ip` command.
:

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

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

```
xe host-list
<!--NeedCopy-->
```

Changing the IP address of the pool coordinator XenServer host requires extra steps. This is because each pool member uses the advertised IP address of the pool coordinator for communication. The pool members do not know how to contact the pool coordinator 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 coordinators.

Use the `pif-reconfigure-ip` CLI command to set the IP address as desired:

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

When the IP address of the pool coordinator changes, all member hosts enter into an emergency mode when they fail to contact the pool coordinator.

On the pool coordinator, use the `pool-recover-slaves` command to force the pool coordinator to contact each pool member and inform them of the new pool coordinator IP address:

```
xe pool-recover-slaves
<!--NeedCopy-->
```

Management interface

When you install XenServer on a host, one of its NICs is designated as the *management interface*: the NIC used for XenServer management traffic. The management interface is used for XenCenter connections to the host (for example, Citrix Virtual Apps and Desktops) and for host-to-host communication.

Use the `pif-list` command to determine which PIF corresponds to the NIC to be used as the management interface. The UUID of each PIF is returned.

```
xe pif-list
<!--NeedCopy-->
```

Use the `pif-param-list` command to verify the IP addressing configuration for the PIF used for the management interface. If necessary, use the `pif-reconfigure-ip` command to configure IP addressing for the PIF to be used.

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

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

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

Use the `network-list` command to determine which PIF corresponds to the NIC to be used as the management interface for all the hosts in the pool. The UUID of pool wide network is returned.

```
xe network-list
<!--NeedCopy-->
```

Use the `network-param-list` command to fetch the PIF UUIDs of all the hosts in the pool. Use the `pif-param-list` command to verify the IP addressing configuration for the PIF for the management interface. If necessary, use the `pif-reconfigure-ip` command to configure IP addressing for the PIF to be used.

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

Use the `pool-management-reconfigure` CLI command to change the PIF used for the management interface listed in the Networks list.

```
xe pool-management-reconfigure network-uuid=network_uuid
<!--NeedCopy-->
```

Restrict use of port 80

You can use either HTTPS over port 443 or HTTP over port 80 to communicate with XenServer. For security reasons, you can close TCP port 80 on the management interface. By default, port 80 is still open. If you close it, any external clients that use the management interface must use HTTPS over port 443 to connect to XenServer. However, before closing port 80, check whether all your API clients (Citrix Virtual Apps and Desktops in particular) can use HTTPS over port 443.

To close port 80, see the `https-only` xe CLI command.

Disable management access

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

> **Warning:**
>
> When the management interface is disabled, you must log in on the physical host console to perform management tasks. External interfaces such as XenCenter do not work when the management interface is disabled.

Add a new physical NIC

1. Install a new physical NIC on your XenServer host in the usual manner.
2. Restart your XenServer host.
3. List all the physical NICs for that XenServer host by using the following command:

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

4. If you do not see the additional NIC, scan for new physical interfaces by using the following command:

   ```
   xe pif-scan host-uuid=<host_uuid>
   ```

   This command creates a new PIF object for the new NIC.
5. List the physical NICs on the XenServer host again to verify that the new NIC is visible:

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

6. The new PIF is initially listed as disconnected (`currently-attached ( RO): false`). To bring it up, use the following command:

   ```
   xe pif-plug uuid=<uuid_of_pif>
   ```

Alternatively, you can use XenCenter to rescan for new NICs. For more information, see Configuring NICs in the XenCenter documentation.

Remove a physical NIC

Before removing the NIC, ensure that you know the UUID of the corresponding PIF. Remove the physical NIC from your XenServer host in the usual manner. After restarting the host, run the xe CLI command `pif-forget uuid=<UUID>` to destroy the PIF object.

Add a purpose to a network

The network purpose can be used to add extra functionalities to a network. For example, the ability to use the network to make NBD connections.

To add a network purpose, use the`xe network-param-add` command:

```
xe network-param-add param-name=purpose param-key=purpose uuid=network-
    uuid
<!--NeedCopy-->
```

To delete a network purpose, use the`xe network-param-remove` command:

```
xe network-param-remove param-name=purpose param-key=purpose uuid=
    network-uuid
<!--NeedCopy-->
```

Currently, the available values for the network purpose are `nbd` and `insecure_nbd`. For more information, see the XenServer Changed Block Tracking Guide.

Use switch port locking

The XenServer switch-port locking feature lets you control traffic sent from unknown, untrusted, or potentially hostile VMs by limiting their ability to pretend they have a MAC or IP address that was not assigned to them. You can use the port-locking commands to block all traffic on a network by default or define specific IP addresses from which an individual VM is allowed to send traffic.

Switch-port locking is a feature designed for public cloud-service providers in environments concerned about internal threats. This functionality assists public cloud-service providers who have a network architecture in which each VM has a public, internet-connected IP address. Because cloud tenants are untrusted, you can use security measures such as spoofing protection to ensure that tenants cannot attack other virtual machines in the cloud.

Using switch-port locking lets you simplify your network configuration by enabling all of your tenants or guests to use the same Layer 2 network.

One of the most important functions of the port-locking commands is they can restrict the traffic that an untrusted guest send. This restricts the guest's ability to pretend it has a MAC or IP address it does not actually possess. Specifically, you can use these commands to prevent a guest from:

- Claiming an IP or MAC address other than the ones the XenServer administrator has specified it can use
- Intercepting, spoofing, or disrupting the traffic of other VMs

Requirements

- The XenServer switch-port locking feature is supported on the Linux bridge and vSwitch networking stacks.
- When you enable Role Based Access Control (RBAC) in your environment, the user configuring switch-port locking must be logged in with an account that has at least a Pool Operator or Pool Admin role. When RBAC is not enabled in your environment, the user must be logged in with the root account for the pool coordinator.
- When you run the switch-port locking commands, networks can be online or offline.
- In Windows guests, the disconnected Network icon only appears when XenServer VM Tools are installed in the guest.

Notes Without any switch-port locking configurations, VIFs are set to "network_default"and Networks are set to "unlocked."

Configuring switch-port locking is not supported when any third-party controllers are in use in the environment.

Switch port locking does not prevent cloud tenants from:

- Performing an IP-level attack on another tenant/user. However, switch-port locking prevents them performing the IP-level attack if they attempt to use the following means to do so and switch-port locking is configured: a) impersonating another tenant in the cloud or user or b) initiating an intercept of traffic intended for another user.
- Exhausting network resources.
- Receiving some traffic intended for other virtual machines through normal switch flooding behaviors (for broadcast MAC addresses or unknown destination MAC addresses).

Likewise, switch-port locking does not restrict where a VM can send traffic to.

Implementation notes You can implement the switch-port locking functionality either by using the command line or the XenServer API. However, in large environments, where automation is a primary concern, the most typical implementation method might be by using the API.

Examples This section provides examples of how switch-port locking can prevent certain types of attacks. In these examples, VM-c is a virtual machine that a hostile tenant (Tenant C) is leasing and using for attacks. VM-a and VM-b are virtual machines leased by non-attacking tenants.

Example 1: How switch port locking can prevent ARP spoofing prevention:

ARP spoofing is used to indicate an attacker's attempts to associate their MAC address with the IP address for another node. ARP spoofing can potentially result in the node's traffic being sent to the attacker instead. To achieve this goal the attacker sends fake (spoofed) ARP messages to an Ethernet LAN.

Scenario:

Virtual Machine A (VM-a) wants to send IP traffic from VM-a to Virtual Machine B (VM-b) by addressing it to VM-b's IP address. The owner of Virtual Machine C wants to use ARP spoofing to pretend their VM, VM-c, is actually VM-b.

1. VM-c sends a speculative stream of ARP replies to VM-a. The ARP replies claim that the MAC address in the reply (c_MAC) is associated with the IP address, b_IP

   Result: Because the administrator enabled switch-port locking, these packets are all dropped because enabling switch-port locking prevents impersonation.

2. VM-b sends an ARP reply to VM-a, claiming that the MAC address in the reply (b_MAC) is associated with the IP address, b_IP.

   Result: VM-a receives VM-b's ARP response.

Example 2: IP Spoofing prevention:

IP address spoofing is a process that conceals the identity of packets by creating Internet Protocol (IP) packets with a forged source IP address.

Scenario:

Tenant C is attempting to perform a Denial of Service attack using their host, Host-C, on a remote system to disguise their identity.

Attempt 1:

Tenant C sets Host-C's IP address and MAC address to VM-a's IP and MAC addresses (a_IP and a_MAC). Tenant C instructs Host-C to send IP traffic to a remote system.

Result: The Host-C packets are dropped. This is because the administrator enabled switch-port locking. The Host-C packets are dropped because enabling switch-port locking prevents impersonation.

Attempt 2:

Tenant C sets Host-C's IP address to VM-a's IP address (a_IP) and keeps their original c_MAC.

Tenant C instructs Host-C to send IP traffic to a remote system.

Result: The Host-C packets are dropped. This is because the administrator enabled switch-port locking, which prevents impersonation.

Example 3: Web hosting:

Scenario:

Alice is an infrastructure administrator.

One of her tenants, Tenant B, is hosting multiple websites from their VM, VM-b. Each website needs a distinct IP address hosted on the same virtual network interface (VIF).

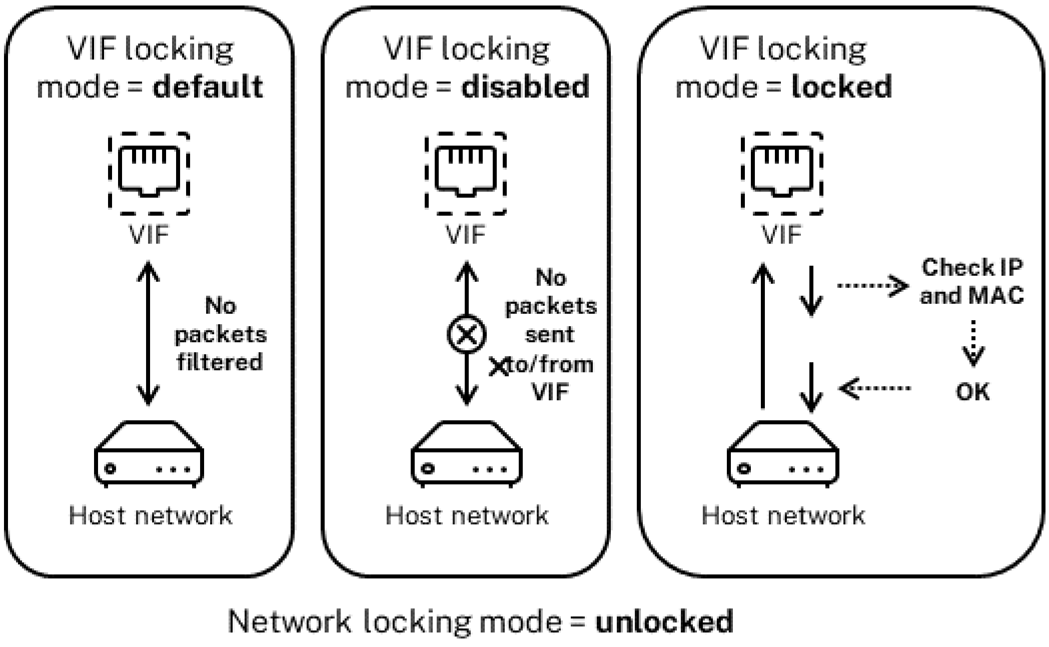
Alice reconfigures Host-B's VIF to be locked to a single MAC but many IP addresses.

How switch-port locking works The switch-port locking feature lets you control packet filtering at one or more of two levels:

- **VIF level**. Settings you configure on the VIF determine how packets are filtered. You can set the VIF to prevent the VM from sending any traffic, restrict the VIF so it can only send traffic using its assigned IP address, or allow the VM to send traffic to any IP address on the network connected to the VIF.
- **Network level**. The XenServer network determines how packets are filtered. When a VIF's locking mode is set to `network_default`, it refers to the network-level locking setting to determine what traffic to allow.

Regardless of which networking stack you use, the feature operates the same way. However, as described in more detail in the sections that follow, the Linux bridge does not fully support switch-port locking in IPv6.

VIF locking-mode states The XenServer switch-port locking feature provides a locking mode that lets you configure VIFs in four different states. These states only apply when the VIF is plugged into a running virtual machine.


VIF locking mode = **default**
VIF
No packets filtered
Host network
VIF locking mode = **disabled**
VIF
No packets sent to/from VIF
Host network
VIF locking mode = **locked**
VIF
Check IP and MAC
OK
Host network
Network locking mode = **unlocked**

- **Network_default**. When the VIF's state is set to `network_default`, XenServer uses the network's `default-locking-mode` parameter to determine if and how to filter packets traveling through the VIF. The behavior varies according to if the associated network has the network default locking mode parameter set to disabled or unlocked:

  -`default-locking-mode=disabled`, XenServer applies a filtering rule so that the VIF drops all traffic.

  -`default-locking-mode`=unlocked, XenServer removes all the filtering rules associated with the VIF. By default, the default locking mode parameter is set to `unlocked`.

  For information about the `default-locking-mode` parameter, see Network commands.

  The default locking mode of the network has no effect on attached VIFs whose locking state is anything other than `network_default`.

  > **Note:**
  >
  > You cannot change the `default-locking-mode` of a network that has active VIFs attached to it.

- **Locked**. XenServer applies filtering rules so that only traffic sent to/from the specified MAC and IP addresses is allowed to be sent out through the VIF. In this mode, if no IP addresses are specified, the VM cannot send any traffic through that VIF, on that network.

  To specify the IP addresses from which the VIF accepts traffic, use the IPv4 or IPv6 IP addresses by using the `ipv4_allowed` or `ipv6_allowed` parameters. However, if you have the Linux bridge configured, do not type IPv6 addresses.

XenServer lets you type IPv6 addresses when the Linux bridge is active. However, XenServer cannot filter based on the IPv6 addresses typed. The reason is the Linux bridge does not have modules to filter Neighbor Discovery Protocol (NDP) packets. Therefore, complete protection cannot be implemented and guests would be able to impersonate another guest by forging NDP packets. As result, if you specify even one IPv6 address, XenServer lets all IPv6 traffic pass through the VIF. If you do not specify any IPv6 addresses, XenServer does not let any IPv6 traffic pass through to the VIF.

- **Unlocked**. All network traffic can pass through the VIF. That is, no filters are applied to any traffic going to or from the VIF.
- **Disabled**. No traffic is allowed to pass through the VIF. (That is, XenServer applies a filtering rule so that the VIF drops all traffic.)

Configure switch port locking This section provides three different procedures:

- Restrict VIFs to use a specific IP address
- Add an IP address to an existing restricted list. For example, to add an IP address to a VIF when the VM is running and connected to the network (for example, if you are taking a network offline temporarily).
- Remove an IP address from an existing restricted list

If a VIF's locking-mode is set to `locked`, it can only use the addresses specified in the `ipv4-allowed` or `ipv6-allowed` parameters.

Because, in some relatively rare cases, VIFs may have more than one IP address, it is possible to specify multiple IP addresses for a VIF.

You can perform these procedures before or after the VIF is plugged in (or the VM is started).

Change the default-locking mode to locked, if it is not using that mode already, by running the following command:

```
xe vif-param-set uuid=vif-uuid locking-mode=locked
<!--NeedCopy-->
```

The `vif-uuid` represents the UUID of the VIF you want to allow to send traffic. To obtain the UUID, run the xe `vif-list` command on the host. `vm-uuid` Indicates the virtual machine for which the information appears. The device ID indicates the device number of the VIF.

Run the `vif-param-set` command to specify the IP addresses from which the virtual machine can send traffic. Do one or more of the following:

- Specify one or more IPv4 IP addresses destinations. For example:

```
xe vif-param-set uuid=vif-uuid ipv4-allowed=comma separated list
    of ipv4-addresses
<!--NeedCopy-->
```

- Specify one or more IPv6 IP addresses destinations. For example:

```
xe vif-param-set uuid=vif-uuid ipv6-allowed=comma separated list
    of ipv6-addresses
<!--NeedCopy-->
```

You can specify multiple IP addresses by separating them with a comma, as shown in the preceding example.

After performing the procedure to restrict a VIF to using a specific IP address, you can add one or more IP addresses the VIF can use.

Run the `vif-param-add` command to add the IP addresses to the existing list. Do one or more of the following:

- Specify the IPv4 IP address. For example:

```
xe vif-param-add uuid=vif-uuid ipv4-allowed=comma separated list
    of ipv4-addresses
<!--NeedCopy-->
```

- Specify the IPv6 IP address. For example:

```
xe vif-param-add uuid=vif-uuid ipv6-allowed=comma separated list
    of ipv6-addresses
<!--NeedCopy-->
```

If you restrict a VIF to use two or more IP addresses, you can delete one of those IP addresses from the list.

Run the `vif-param-remove` command to delete the IP addresses from the existing list. Do one or more of the following:

- Specify the IPv4 IP address to delete. For example:

```
xe vif-param-remove uuid=vif-uuid ipv4-allowed=comma separated
    list of ipv4-addresses
<!--NeedCopy-->
```

- Specify the IPv6 IP address to delete. For example:

```
xe vif-param-remove uuid=vif-uuid ipv6-allowed=comma separated
    list of ipv6-addresses
<!--NeedCopy-->
```

Prevent a virtual machine from sending or receiving traffic from a specific network The following procedure prevents a virtual machine from communicating through a specific VIF. As a VIF connects to a specific XenServer network, you can use this procedure to prevent a virtual machine from sending or receiving any traffic from a specific network. This provides a more granular level of control than disabling an entire network.

If you use the CLI command, you do not need to unplug the VIF to set the VIF's locking mode. The command changes the filtering rules while the VIF is running. In this case, the network connection still appears to be present, however, the VIF drops any packets the VM attempts to send.

> **Tip:**
>
> To find the UUID of a VIF, run the xe `vif-list` command on the host. The device ID indicates the device number of the VIF.

To prevent a VIF from receiving traffic, disable the VIF connected to the network from which you want to stop the VM from receiving traffic:

```
xe vif-param-set uuid=vif-uuid locking-mode=disabled
<!--NeedCopy-->
```

You can also disable the VIF in XenCenter by selecting the virtual network interface in the VM's Networking tab and clicking Deactivate.

Remove a VIF's restriction to an IP address To revert to the default (original) locking mode state, use the following procedure. By default, when you create a VIF, XenServer configures it so that it is not restricted to using a specific IP address.

To revert a VIF to an unlocked state, change the VIF default-locking mode to unlocked. If it is not using that mode already, run the following command:

```
xe vif-param-set uuid=vif_uuid locking-mode=unlocked
<!--NeedCopy-->
```

Simplify VIF locking mode configuration in the Cloud Rather than running the VIF locking mode commands for each VIF, you can ensure all VIFs are disabled by default. To do so, you must change the packet filtering at the network level. Changing the packet filtering causes the XenServer network to determine how packets are filtered, as described in the previous section *How switch-port locking works*.

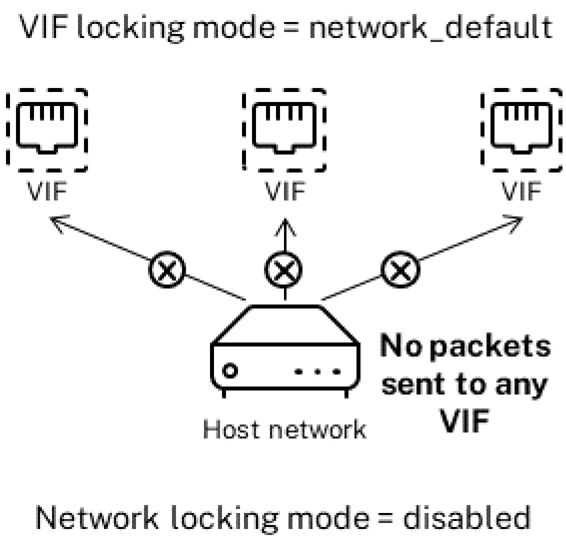
Specifically, a network's **`default-locking-mode`** setting determines how new VIFs with default settings behave. Whenever a VIF's `locking-mode` is set to **`default`**, the VIF refers to the network-locking mode (**`default-locking-mode`**) to determine if and how to filter packets traveling through the VIF:

- **Unlocked**. When the network **default**-locking-mode parameter is set to unlocked, XenServer lets the VM send traffic to any IP address on the network the VIF connects to.
- **Disabled**. When the **default**-locking-mode parameter is set to disabled, XenServer applies a filtering rule so that the VIF drops all traffic.

By default, the **default**-locking-mode for all networks created in XenCenter and using the CLI are set to unlocked.

By setting the VIF's locking mode to its default (network_default), you can create a basic default configuration (at the network level) for all newly created VIFs that connect to a specific network.

This illustration shows how, when a VIF's locking-mode is set to its default setting (network_default), the VIF uses the network **default**-locking-mode to determine its behavior.


VIF locking mode = network_default
VIF
VIF
VIF
No packets sent to any VIF
Host network
Network locking mode = disabled

For example, by default, VIFs are created with their locking-mode set to network_default. If you set a network's **default**-locking-mode=disabled, any new VIFs for which you have not configured the locking mode are disabled. The VIFs remain disabled until you either (a) change the individual VIF's locking-mode parameter or (b) explicitly set the VIF's locking-mode to 'unlocked. This is helpful when you trust a specific VM enough so you do not want to filter its traffic at all.

To change a network's default locking mode setting:

After creating the network, change the default-locking mode by running the following command:

```
xe network-param-set uuid=network-uuid default-locking-mode=[unlocked|
    disabled]
<!--NeedCopy-->
```

> **Note:**
>
> To get the UUID for a network, run the xe `network-list` command. This command displays the UUIDs for all the networks on the host on which you ran the command.

To check a network's default locking mode setting:

Run one of the following commands:

```
xe network-param-get uuid=network-uuid param-name=default-locking-mode
<!--NeedCopy-->
```

OR

```
xe network-list uuid=network-uuid params=default-locking-mode
<!--NeedCopy-->
```

Use network settings for VIF traffic filtering The following procedure instructs a VIF on a virtual machine to use the XenServer network `default-locking-mode` settings on the network itself to determine how to filter traffic.

1. Change the VIF locking state to `network_default`, if it is not using that mode already, by running the following command:

   ```
   xe vif-param-set uuid=vif_uuid locking-mode=network_default
   <!--NeedCopy-->
   ```

2. Change the default-locking mode to `unlocked`, if it is not using that mode already, by running the following command:

   ```
   xe network-param-set uuid=network-uuid default-locking-mode=
       unlocked
   <!--NeedCopy-->
   ```

Troubleshoot networking

January 31, 2024

If you are experiencing problems with configuring networking, first ensure that you have not directly changed any of the control domain `ifcfg-*` files. The control domain host agent manages the `ifcfg` files directly, and any changes are 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 see corrupted traffic to VMs, try to obtain the latest firmware from your vendor and then use it to update your hardware.

If the problem still persists, then you can use the CLI to disable receive or transmit offload optimizations on the physical interface.

> **Warning:**
>
> Disabling receive or transmit offload optimizations can result in a performance loss and 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
<!--NeedCopy-->
```

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

```
xe pif-param-set uuid=pif_uuid other-config:ethtool-tx=off
<!--NeedCopy-->
```

Finally, replug the PIF or restart the host for the change to take effect.

Emergency network reset

Incorrect networking settings can cause loss of network connectivity. When there is no network connectivity, XenServer host can become inaccessible through XenCenter or remote SSH. Emergency Network Reset provides a simple mechanism to recover and reset a host's networking.

The Emergency network reset feature is available from the CLI using the `xe-reset-networking` command, and within the **Network and Management Interface** section of `xsconsole`.

Incorrect settings that cause a loss of network connectivity include renaming network interfaces, creating bonds or VLANs, or mistakes when changing the management interface. For example, typing the wrong IP address. You may also want to run this utility in the following scenarios:

- When a rolling pool upgrade, manual upgrade, hotfix installation, or driver installation causes a lack of network connectivity, or
- If a pool coordinator or host in a resource pool is unable to contact with other hosts.

Use the `xe-reset-networking` utility only in an emergency because it deletes the configuration for all PIFs, bonds, VLANs, and tunnels associated with the host. Guest Networks and VIFs are preserved. As part of this utility, VMs are shut down forcefully. Before running this command, cleanly shut down the VMs where possible. Before you apply a reset, you can change the management interface and specify which IP configuration, DHCP, or Static can be used.

If the pool coordinator requires a network reset, reset the network on the pool coordinator first before applying a network reset on pool members. Apply the network reset on all remaining hosts in the pool to ensure that the pool's networking configuration is homogeneous. Network homogeneity is an important factor for live migration.

> **Note:**
>
> If the pool coordinator's IP address (the management interface) changes as a result of a network reset or `xe host-management-reconfigure`, apply the network reset command to other hosts in the pool. This is to ensure that the pool members can reconnect to the pool coordinator on its new IP address. In this situation, the IP address of the pool coordinator must be specified.
>
> Network reset is NOT supported when High Availability is enabled. To reset network configuration in this scenario, you must first manually disable high availability, and then run the network reset command.

Verifying the network reset

After you specify the configuration mode to be used after the network reset, `xsconsole` and the CLI display settings that will be applied after host reboot. It is a final chance to modify before applying the emergency network reset command. After restart, the new network configuration can be verified in XenCenter and `xsconsole`. In XenCenter, with the host selected, select the **Networking** tab to see the new network configuration. The Network and Management Interface section in **xsconsole** display this information.

> **Note:**
>
> Run emergency network reset on other pool members to replicate bonds, VLANs, or tunnels from the pool coordinator's new configuration.

Using the CLI for network reset

The following table shows the available optional parameters which can be used by running the `xe-reset-networking` command.

> **Warning:**
>
> Users are responsible to ensure the validity of parameters for the `xe-reset-networking` command, and to check the parameters carefully. If you specify invalid parameters, network connectivity and configuration can be lost. In this situation, we advise that you rerun the command `xe-reset-networking` without using any parameters.

> Resetting the networking configuration of a whole pool **must** begin on the pool coordinator, followed by network reset on all remaining hosts in the pool.

Parameter	Required/Optional	Description
`-m`, `--master`	Optional	IP address of the pool coordinator's management interface. Defaults to the last known pool coordinator's IP address.
`--device`	Optional	Device name of the management interface. Defaults to the device name specified during installation.
`--mode=static`	Optional	Enables the following four networking parameters for static IP configuration for the management interface. If not specified, networking is configured using DHCP.
`--ip`	Required, if `mode=static`	IP address for the host's management interface. Only valid if `mode=static`.
`--netmask`	Required, if `mode=static`	Netmask for the management interface. Only valid if `mode=static`.
`--gateway`	Optional	Gateway for the management interface. Only valid if `mode=static`.
`--dns`	Optional	DNS Server for the management interface. Only valid if `mode=static`.
`--vlan`	Optional	VLAN tag for the management interface. Defaults to the VLAN tag specified during installation.

Pool cooordinator command-line examples Examples of commands that can be applied on a pool coordinator:

To reset networking for DHCP configuration:

```
xe-reset-networking
<!--NeedCopy-->
```

To reset networking for Static IP configuration:

```
xe-reset-networking --mode= static --ip=ip-address \
    --netmask=netmask --gateway=gateway \
    --dns=dns
<!--NeedCopy-->
```

To reset networking for DHCP configuration if another interface became the management interface after initial setup:

```
xe-reset-networking --device=device-name
<!--NeedCopy-->
```

To reset networking for Static IP configuration if another interface became the management interface after initial setup:

```
xe-reset-networking --device=device-name --mode=static \
    --ip=ip-address --netmask=netmask \
    --gateway=gateway --dns=dns
<!--NeedCopy-->
```

To reset networking for management interface on VLAN:

```
xe-reset-networking --vlan=VLAN TAG
<!--NeedCopy-->
```

> **Note:**
>
> The `reset-network` command can also be used along with the IP configuration settings.

Pool member command-line examples All previous examples also apply to pool members. Additionally, the pool coordinator's IP address can be specified (which is necessary if it has changed.)

To reset networking for DHCP configuration:

```
xe-reset-networking
<!--NeedCopy-->
```

To reset networking for DHCP if the pool coordinator's IP address was changed:

```
xe-reset-networking --master=pool-coordinator-ip-address
<!--NeedCopy-->
```

To reset networking for Static IP configuration, assuming the pool coordinator's IP address didn't change:

```
xe-reset-networking --mode=static --ip=ip-address --netmask=netmask \
    --gateway=gateway --dns=dns
<!--NeedCopy-->
```

To reset networking for DHCP configuration if the management interface and the pool coordinator's IP address was changed after initial setup:

```
xe-reset-networking --device=device-name --master=pool-coordinator-ip-
   address
<!--NeedCopy-->
```

Storage

February 7, 2024

This section describes how physical storage hardware maps to virtual machines (VMs), and the software objects used by the management API to perform storage-related tasks. Detailed sections on each of the supported storage types include the following information:

- Procedures for creating storage for VMs using the CLI, with type-specific device configuration options
- Generating snapshots for backup purposes
- Best practices for managing storage

Storage repositories (SRs)

A Storage Repository (SR) is a particular storage target, in which Virtual Machine (VM) Virtual Disk Images (VDIs) are stored. A VDI is a storage abstraction that represents a virtual hard disk drive (HDD).

SRs are flexible, with built-in support for the following drives:

Locally connected:

- SATA
- SCSI
- SAS
- NVMe

The local physical storage hardware can be a hard disk drive (HDD) or a solid state drive (SSD).

Remotely connected:

- iSCSI

- NFS
- SAS
- SMB (version 3 only)
- Fibre Channel

> **Note:**
>
> NVMe over Fibre Channel and NVMe over TCP are not supported.

The SR and VDI abstractions allow for advanced storage features to be exposed on storage targets that support them. For example, advanced features such as *thin provisioning*, VDI snapshots, and fast cloning. For storage subsystems that don't support advanced operations directly, a software stack that implements these features is provided. This software stack is based on Microsoft's Virtual Hard Disk (VHD) specification.

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

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 in the pool. All hosts in a single resource pool must have at least one shared SR in common. Shared storage cannot be shared between multiple pools.

SR commands provide operations for creating, destroying, resizing, cloning, connecting and discovering the individual VDIs that they contain. CLI operations to manage storage repositories are described in SR commands.

> **Warning:**
>
> XenServer does not support snapshots at the external SAN-level of a LUN for any SR type.

Virtual disk image (VDI)

A virtual disk image (VDI) is a storage abstraction that represents a virtual hard disk drive (HDD). VDIs are the fundamental unit of virtualized storage in XenServer. VDIs are persistent, on-disk objects that exist independently of XenServer hosts. CLI operations to manage VDIs are described in VDI commands. The on-disk representation of the data differs by SR type. A separate storage plug-in interface for each SR, called the SM API, manages the data.

Physical block devices (PBDs)

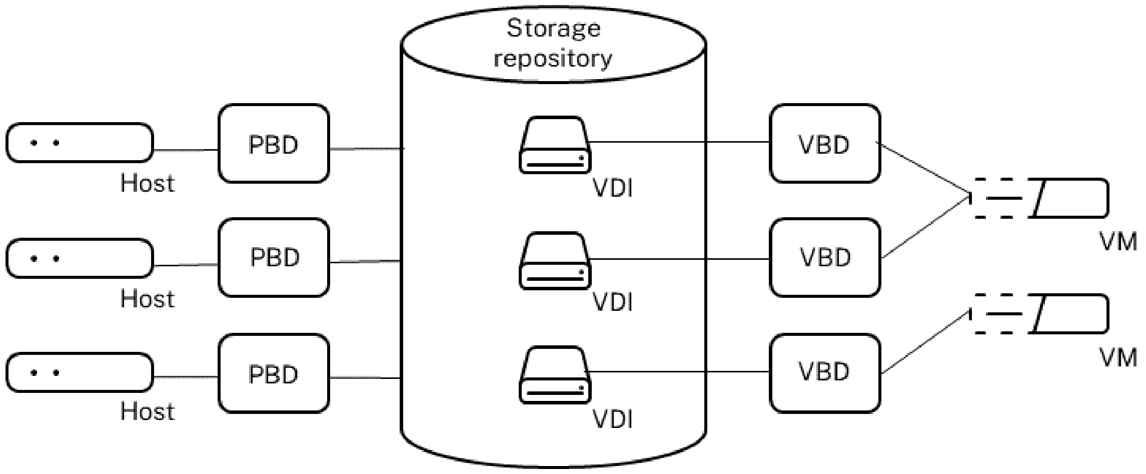
Physical block devices represent the interface between a physical server and an attached SR. PBDs are connector objects that allow a given SR to be mapped to a host. PBDs store the device configuration fields that are used to connect to and interact with a given storage target. For example, NFS device 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 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 a VDI into a VM, VBDs allow for the fine-tuning of parameters regarding the disk I/O priority and statistics of a given VDI, and whether that VDI can be booted. CLI operations relating to VBDs are described in VBD commands.

Summary of storage objects

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



Virtual disk data formats

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

1. *Logical volume-based VHD on a LUN:* The default XenServer block-based storage inserts a logical volume manager on a disk. This disk is either a locally attached device (LVM) or a SAN attached

LUN over either Fibre Channel, iSCSI, or SAS. 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.

2. *File-based QCOW2 on a LUN:* VM images are stored as thin-provisioned QCOW2 format files on a GFS2 shared-disk filesystem on a LUN attached over either iSCSI software initiator or Hardware HBA.
3. *File-based VHD on a filesystem:* VM images are stored as thin-provisioned VHD format files on either a local non-shared filesystem (EXT3/EXT4 type SR), a shared NFS target (NFS type SR), or a remote SMB target (SMB type SR).
4. *File-based QCOW2 on a filesystem:* VM images are stored as thin-provisioned QCOW2 format files on a local non-shared XFS filesystem.

VDI types

For GFS2 and XFS SRs, QCOW2 VDIs are created.

For other SR types, VHD format VDIs are created. You can opt to use raw at the time you create the VDI. This option can only be specified by using the xe CLI.

> **Note:**
>
> If you create a raw VDI on an LVM-based SR or HBA/LUN-per-VDI SR, it might allow the owning VM to access data that was part of a previously deleted VDI (of any format) belonging to any VM. We recommend that you consider your security requirements before using this option.
>
> Raw VDIs on a NFS, EXT, or SMB SR do not allow access to the data of previously deleted VDIs belonging to any VM.

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.

Create a raw virtual disk by 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
<!--NeedCopy-->
```

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

Convert between VDI formats

It is not possible to do a direct conversion between the raw and VHD formats. Instead, you can create a VDI (either raw, as described above, or VHD) and then copy data into it from an existing volume. Use the xe CLI to ensure that the new VDI has a virtual size at least as large as the VDI you are copying from. You can do this 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 to do a direct block-copy of the data. For example, standard disk management tools in Windows or the `dd` command in Linux. If the new volume is a VHD volume, use a tool that can avoid writing empty sectors to the disk. This action can ensure that space is used optimally in the underlying storage repository. A file-based copy approach may be more suitable.

VHD-based and QCOW2-based VDIs

VHD and QCOW2 images can be *chained*, allowing two VDIs to share common data. In cases where a VHD-backed or QCOW2-backed VM is cloned, the resulting VMs share the common on-disk data at the time of cloning. Each VM proceeds to make its own changes in an isolated copy-on-write version of the VDI. This feature allows such VMs to be quickly cloned from templates, facilitating very fast provisioning and deployment of new VMs.

As VMs and their associated VDIs get cloned over time this creates trees of chained VDIs. 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 time taken to perform the process depends on the size of the VDI and amount of shared data.

Both the VHD and QCOW2 formats support *thin provisioning*. The image file is automatically extended in fine granular chunks as the VM writes data into the disk. For file-based VHD and GFS2-based QCOW2, this approach 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 copy-on-write instance disk is reclaimed when a snapshot or clone occurs. The difference between the two behaviors can be described in the following way:

- For *LVM-based VHD images*, the difference disk nodes within the chain consume only as much data as has been written to disk. However, 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 are fully inflated on attach, and deflated on detach.

- For *file-based VHDs* and *GFS2-based QCOW2 images*, all nodes consume only as much data as has been written. The leaf node files grow to accommodate data as it is actively written. If a 100 GB VDI is allocated for a VM and an OS is installed, the VDI file is physically only the size of the OS data on the disk, plus some minor metadata overhead.

When cloning VMs based on a single VHD or QCOW2 template, each child VM forms a chain where new changes are written to the new VM. 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 results in degraded performance. XenServer supports a maximum chain length of 30. Do not approach this limit without good reason. If in doubt, "copy"the VM using XenCenter or use the `vm-copy` command, which resets the chain length back to 0.

VHD-specific notes on coalesce Only one coalescing process is ever active for an SR. This process thread runs on the SR pool coordinator.

If you have critical VMs running on the pool coordinator, you can take the following steps to mitigate against occasional slow I/O:

- Migrate the VM to a host other than the SR pool coordinator
- Set the disk I/O priority to a higher level, and adjust the scheduler. For more information, see Virtual disk I/O request prioritization.

Create a storage repository

April 23, 2024

You can use the **New Storage Repository** wizard in XenCenter to create storage repositories (SRs). The wizard guides you through the configuration steps. Alternatively, use the CLI, and the `sr-create` command. The `sr-create` command creates an SR on the storage substrate (potentially destroying any existing data). It also 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 in the resource pool.

If you are creating an SR for IP-based storage (iSCSI or NFS), you can configure one of the following as the storage network: the NIC that handles the management traffic or a new NIC for the storage traffic. To assign an IP address to a NIC, see Configure a dedicated storage NIC.

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 (EXT3/EXT4, NFS, GFS2) support full thin provisioning, including for virtual disks that are active.

Warnings:

- When VHD VDIs are not attached to a VM, for example for a VDI snapshot, they are stored as thinly provisioned by default. If you attempt to reattach the VDI, ensure that there is sufficient disk-space available for the VDI to become thickly provisioned. VDI clones are thickly provisioned.
- XenServer does not support snapshots at the external SAN-level of a LUN for any SR type.
- Do not attempt to create an SR where the LUN ID of the destination LUN is greater than 255. Ensure that your target exposes the LUN with a LUN ID that is less than or equal to 255 before using this LUN to create an SR.
- If you use thin provisioning on a file-based SR, ensure that you monitor the free space on your SR. If the SR usage grows to 100%, further writes from VMs fail. These failed writes can cause the VM to freeze or crash.

The maximum supported VDI sizes are:

Storage Repository Format	Maximum VDI size
EXT3/EXT4	2 TiB
GFS2 (with iSCSI or HBA)	16 TiB
XFS	16 TiB
LVM	2 TiB
LVMoFCOE (deprecated)	2 TiB
LVMoHBA	2 TiB
LVMoiSCSI	2 TiB
NFS	2 TiB
SMB	2 TiB

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.

> **Note:**
>
> The block size of an LVM LUN must be 512 bytes. To use storage with 4 KB physical blocks, the storage must also support emulation of 512 byte allocation blocks (the logical block size must be 512 bytes).

LVM performance considerations

The snapshot and fast clone functionality for LVM-based SRs comes with an inherent performance overhead. When optimal performance is required, 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.

> **Warning:**
>
> Do not try to snapshot a VM that has `type=raw` disks attached. This action can 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.

Creating a local LVM SR

An LVM SR is created by default on host install.

Device-config parameters for LVM SRs are:

Parameter Name	Description	Required?
`device`	Device name on the local host to use for the SR. You can also provide a comma-separated list of names.	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
<!--NeedCopy-->
```

Local EXT3/EXT4

Using EXT3/EXT4 enables thin provisioning on local storage. However, the default storage repository type is LVM as it gives a consistent write performance and, prevents storage over-commit. If you use EXT3/EXT4, you might see reduced performance in the following cases:

- When carrying out VM lifecycle operations such as VM create and suspend/resume
- When creating large files from within the VM

Local disk EXT3/EXT4 SRs must be configured using the XenServer CLI.

Whether a local EXT SR uses EXT3 or EXT4 depends on what version of XenServer created it:

- If you created the local EXT SR on an earlier version of Citrix Hypervisor or XenServer and then upgraded to XenServer 8, it uses EXT3.
- If you created the local EXT SR on XenServer 8, it uses EXT4.

> **Note:**
>
> The block size of an EXT3/EXT4 disk must be 512 bytes. To use storage with 4 KB physical blocks, the storage must also support emulation of 512 byte allocation blocks (the logical block size must be 512 bytes).

Creating a local EXT4 SR (`ext`)

Device-config parameters for EXT SRs:

Parameter Name	Description	Required?
`device`	Device name on the local host to use for the SR. You can also provide a comma-separated list of names.	Yes

To create a local EXT4 SR on `/dev/sdb`, use the following command:

```
    xe sr-create host-uuid=valid_uuid content-type=user \
       name-label="Example Local EXT4 SR" shared=false \
       device-config:device=/dev/sdb type=ext
<!--NeedCopy-->
```

Local XFS

Using XFS enables thin provisioning on local storage. The local XFS type allows you to create local storage devices with 4 KB physical blocks without requiring a logical block size of 512 bytes.

Creating a local XFS SR

Device-config parameters for XFS SRs:

Parameter Name	Description	Required?
`device`	Device name on the local host to use for the SR. You can also provide a comma-separated list of names.	Yes

To create a local XFS SR on `/dev/sdb`, use the following command:

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

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.

The following ISO SR types are available:

- `nfs_iso`: The NFS ISO SR type handles CD images stored as files in ISO format available as an NFS share.
- `cifs`: The Windows File Sharing (SMB/CIFS) SR type handles CD images stored as files in ISO format available as a Windows (SMB/CIFS) share.

If you do not specify the storage type to use for the SR, XenServer uses the `location` device config parameter to decide the type.

Device-config parameters for ISO SRs:

Parameter Name	Description	Required?
`location`	Path to the mount.	Yes
`type`	Storage type to use for the SR: `cifs` or `nfs_iso`.	No
`nfsversion`	For the storage type NFS, the version of the NFS protocol to use: 3, 4, 4.0, or 4.1.	No
`vers`	For the storage type CIFS/SMB, the version of SMB to use: 1.0 or 3.0. The default is 3.0.	No
`username`	For the storage type CIFS/SMB, if a username is required for the Windows file server.	No
`cifspassword_secret`	(Recommended) For the storage type CIFS/SMB, you can pass a secret instead of a password for the Windows file server.	No
`cifspassword`	For the storage type CIFS/SMB, if a password is required for the Windows file server. We recommend you use the `cifspassword_secret` parameter instead.	No

> **Note:**
>
> When running the `sr-create` command, we recommend that you use the `device-config:cifspassword_secret` argument instead of specifying the password on the command line. For more information, see Secrets.

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=<path_to_mount> device-config:type=nfs_iso
<!--NeedCopy-->
```

You can use NFS or SMB to mount the ISO SR. For more information about using these SR types, see NFS and SMB.

We recommend that you use SMB version 3 to mount ISO SR on Windows file server. Version 3 is selected by default because it is more secure and robust than SMB version 1.0. However, you can mount ISO SR using SMB version 1 using the following command:

```
    xe sr-create content-type=iso type=iso shared=true device-config:
       location=<path_to_mount>
    device-config:username=<username> device-config:cifspassword=<
       password> \
    device-config:type=cifs device-config:vers=1.0 name-label="Example
        ISO SR"
<!--NeedCopy-->
```

Software iSCSI support

XenServer supports 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 the steps for Fibre Channel HBAs. Both sets of steps are described in Create a Shared LVM over Fibre Channel / Fibre Channel over Ethernet / iSCSI HBA or SAS SR.

Shared iSCSI support using the software iSCSI initiator is implemented based on the Linux Volume Manager (LVM). This feature provides the same performance benefits provided by LVM VDIs in the local disk case. Shared iSCSI SRs using the software-based host initiator can support VM agility using live migration: 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.

> **Note:**
>
> The block size of an iSCSI LUN must be 512 bytes. To use storage with 4 KB physical blocks, the storage must also support emulation of 512 byte allocation blocks (the logical block size must be 512 bytes).

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 names 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. All iSCSI targets/LUNs that your XenServer host accesses 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 typically default to restricting LUN access to a single initiator to ensure data integrity. If an iSCSI LUN is used as a shared SR across multiple hosts in a 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
<!--NeedCopy-->
```

> **Warning:**
>
> - Each iSCSI target and initiator must have a unique IQN. If a non-unique IQN identifier is used, data corruption or denial of LUN access can occur.
> - Do not change the XenServer host IQN with iSCSI SRs attached. Doing so can result in failures connecting to new targets or existing SRs.

Software FCoE storage (deprecated)

Software FCoE provides a standard framework to which hardware vendors can plug in their FCoE-capable NIC and get the same benefits of a hardware-based FCoE. This feature eliminates the need for using expensive HBAs.

> **Note:**
>
> Software FCoE is deprecated and will be removed in a future release.

Before you create a software FCoE storage, manually complete the configuration required to expose a LUN to the host. This configuration includes configuring the FCoE fabric and allocating LUNs to your SAN's public world wide name (PWWN). After you complete this configuration, the available LUN is mounted to the host's CNA as a SCSI device. The SCSI device can then be used to access the LUN as if it were a locally attached SCSI device. For information about configuring the physical switch and the array to support FCoE, see the documentation provided by the vendor.

> **Note:**
>
> Software FCoE can be used with Open vSwitch and Linux bridge as the network back-end.

Create a Software FCoE SR

Before creating a Software FCoE SR, customers must ensure that there are FCoE-capable NICs attached to the host.

Device-config parameters for FCoE SRs are:

Parameter Name	Description	Required?
`SCSIid`	The SCSI bus ID of the destination LUN	Yes

Run the following command to create a shared FCoE SR:

```
    xe sr-create type=lvmofcoe \
    name-label="FCoE SR" shared=true device-config:SCSIid=SCSI_id
<!--NeedCopy-->
```

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 details on configuring QLogic Fibre Channel and iSCSI HBAs, see the Cavium 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
   <!--NeedCopy-->
   ```

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

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

3. Use the xe `sr-probe` command to force a rescan of the HBA controller and display available LUNs. For more information, see Probe an SR and Create a Shared LVM over Fibre Channel / Fibre Channel over Ethernet / iSCSI HBA or SAS SR.

Remove HBA-based SAS, FC, or iSCSI device entries

> **Note:**
>
> This step is not required. We recommend 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 Remove SRs 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. For more information, Probe an SR.
4. Remove the device entries with the following command:

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

> **Warning:**
>
> Make sure that you are certain which LUN you are removing. Accidentally removing a LUN required for host operation, such as the boot or root device, renders the host unusable.

Shared LVM storage

The Shared LVM type represents disks as Logical Volumes within a Volume Group created on an iSCSI (FC or SAS) LUN.

> **Note:**
>
> The block size of an iSCSI LUN must be 512 bytes. To use storage with 4 KB physical blocks, the storage must also support emulation of 512 byte allocation blocks (the logical block size must be 512 bytes).

Create a shared LVM over iSCSI SR by using the Software iSCSI initiator

Device-config parameters for LVMoiSCSI SRs:

Parameter Name	Description	Required?
`target`	The IP address or host name of the iSCSI filer that hosts the SR. This can also be a comma-separated list of values.	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 user name to be used for CHAP authentication	No
`chappassword_secret`	(Recommended) Secret ID for the password to be used for CHAP authentication. Pass a secret instead of a password.	No
`chappassword`	The password to be used for CHAP authentication. We recommend you use the `chappassword_secret` parameter instead.	No
`port`	The network port number on which to query the target	No
`usediscoverynumber`	The specific iSCSI record index to use	No
`incoming_chapuser`	The user name that the iSCSI filter uses to authenticate against the host	No
`incoming_chappassword`	The password that the iSCSI filter uses to authenticate against the host	No

> **Note:**
>
> When running the `sr-create` command, we recommend that you use the `device-config:chappassword_secret` argument instead of specifying the password on the command line.

> For more information, see Secrets.

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
<!--NeedCopy-->
```

Create a Shared LVM over Fibre Channel / Fibre Channel over Ethernet / iSCSI HBA or SAS SR

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. For more information, see your SAN documentation.
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`

   For an example of QLogic iSCSI HBA configuration, see *Hardware host bus adapters (HBAs)* in the previous section. For more information on Fibre Channel and iSCSI HBAs, see the Broadcom and Cavium websites.
3. Use the `sr-probe` command to determine the global device path of the HBA LUN. The `sr-probe` command forces a rescan of HBAs installed in the system to detect any new LUNs that have been zoned to the host. The command returns a list of properties for each LUN found. Specify the `host-uuid` parameter to ensure that the probe occurs on the desired host.

   The global device path returned as the <`path`> property is common across all hosts in the pool. Therefore, this path 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 from 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-360
                a9800068666949673446387665336f
        </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>
<!--NeedCopy-->
```

4. On the pool coordinator, create the SR. Specify the global device path returned in the `<path>` property from `sr-probe`. PBDs are 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
<!--NeedCopy-->
```

> **Note:**
>
> You can use the XenCenter Repair Storage Repository function to retry the PBD creation and plugging portions of the `sr-create` operation. This function 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.

Thin-provisioned shared GFS2 block storage

Thin provisioning better utilizes the available storage by allocating disk storage space to VDIs as data is written to the virtual disk, rather than allocating the full virtual size of the VDI in advance. Thin provisioning enables you to significantly reduce the amount of space required on a shared storage array, and with that your Total Cost of Ownership (TCO).

Thin provisioning for shared block storage is of particular interest in the following cases:

- You want increased space efficiency. Images are sparsely and not thickly allocated.
- You want to reduce the number of I/O operations per second on your storage array. The GFS2 SR is the first SR type to support storage read caching on shared block storage.
- You use a common base image for multiple virtual machines. The images of individual VMs will then typically utilize even less space.
- You use snapshots. Each snapshot is an image and each image is now sparse.
- Your storage does not support NFS and only supports block storage. If your storage supports NFS, we recommend you use NFS instead of GFS2.
- You want to create VDIs that are greater than 2 TiB in size. The GFS2 SR supports VDIs up to 16 TiB in size.

> **Note:**
>
> We recommend not to use a GFS2 SR with a VLAN due to a known issue where you cannot add or remove hosts on a clustered pool if the cluster network is on a non-management VLAN.

The shared GFS2 SR type creates a GFS2 filesystem on an iSCSI or HBA LUN. VDIs are stored in the GFS2 SR as files in the QCOW2 image format.

For more information about using GFS2 storage, see Thin-provisioned shared GFS2 block storage.

NFS and SMB

Shares on NFS servers (that support any version of NFSv4 or NFSv3) or on SMB servers (that support SMB 3) can be used immediately as an SR for virtual disks. VDIs are stored in the Microsoft VHD format only. Additionally, as these SRs can be shared, VDIs stored on shared SRs allow:

- VMs to be started on any XenServer hosts in a resource pool
- VM migrate between XenServer hosts in a resource pool using live migration (without noticeable downtime)

> **Important:**
>
> - Support for SMB3 is limited to the ability to connect to a share using the 3 protocol. Extra features like Transparent Failover depend on feature availability in the upstream Linux kernel and are not supported in XenServer 8.
> - Clustered SMB is not supported with XenServer.
> - For NFSv4, only the authentication type `AUTH_SYS` is supported.
> - SMB storage is available for XenServer Premium Edition customers.
> - It is highly recommended for both NFS and SMB storage that a dedicated storage network be used, using at least two bonded links, ideally to independent network switches with redundant power supplies.
> - When using SMB storage, do not remove the share from the storage before detaching the SMB SR.

VDIs stored on file-based SRs are *thinly provisioned*. The image file is allocated as the VM writes data into the disk. This approach has the considerable benefit that the VM image files take up only as much space on the storage as is required. For example, if a 100 GB VDI is allocated for a VM and an OS is installed, the VDI file only reflects the size of the OS data written to the disk rather than the entire 100 GB.

VHD files may also be chained, allowing two VDIs to share common data. In cases where a file-based VM is cloned, the resulting VMs share the common on-disk data at the time of cloning. Each VM proceeds to make its own changes in an isolated copy-on-write version of the VDI. This feature allows file-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.

File-based SRs and VHD implementations in XenServer assume that they have full control over the SR directory on the file server. Administrators must not modify the contents of the SR directory, as this action can risk corrupting the contents of VDIs.

XenServer has been tuned for enterprise-class storage that uses 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 FAS2020 and FAS3210 storage, using Data OnTap 7.3 and 8.1

> **Warning:**
>
> As VDIs on file-based SRs are created as thin provisioned, administrators must ensure that the file-based SRs have enough disk space for all required VDIs. XenServer hosts do not enforce that the space required for VDIs on file-based SRs is present.
>
> Ensure that you monitor the free space on your SR. If the SR usage grows to 100%, further writes from VMs fail. These failed writes can cause the VM to freeze or crash.

Create a shared NFS SR (NFS)

> **Note:**
>
> If you attempt to attach a read-only NFS SR, this action fails with the following error message: "SR_BACKEND_FAILURE_461 - The file system for SR cannot be written to."

To create an NFS SR, you must provide the hostname or IP address of the NFS server. You can create the SR on any valid destination path; use the `sr-probe` command to display a list of valid destination paths exported by the server.

In scenarios where XenServer is used with lower-end storage, it cautiously waits for all writes to be acknowledged before passing acknowledgments on to VMs. This approach incurs a noticeable performance cost, and might be solved 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. Consider the risks of failure carefully in these situations.

> **Note:**
>
> The NFS server must be configured to export the specified path to all hosts in the pool. If this configuration is not done, the creation of the SR and the plugging of the PBD record fails.

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

The following `device-config` parameters are used with 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
`nfsversion`	Specifies the version of NFS to use. If you specify `nfsversion="4"`, the SR uses NFS v4.0, v4.1 or v4.2, depending on what is available. If you want to select a more specific version of NFS, you can specify `nfsversion="4.0"` and so on. Only one value can be specified for `nfsversion`.	No
`useUDP`	Configure the SR to use UDP rather than the default TCP.	No

For example, to create a shared NFS SR on `192.168.1.10:/export1`, using any version 4 of NFS that is made available by the filer, use the following command:

```
    xe sr-create content-type=user \
    name-label="shared NFS SR" shared=true \
    device-config:server=192.168.1.10 device-config:serverpath=/export1
        type=nfs \
    device-config:nfsversion="4"
<!--NeedCopy-->
```

To create a non-shared NFS SR on `192.168.1.10:/export1`, using specifically NFS version 4.0, run the following command:

```
    xe sr-create host-uuid=host_uuid content-type=user \
    name-label="Non-shared NFS SR" \
    device-config:server=192.168.1.10 device-config:serverpath=/export1
        type=nfs \
    device-config:nfsversion="4.0"
<!--NeedCopy-->
```

Create a shared SMB SR (SMB)

To create an SMB SR, provide the hostname or IP address of the SMB server, the full path of the exported share, and appropriate credentials.

Device-config parameters for SMB SRs:

Parameter Name	Description	Required?
`server`	Full path to share on server	Yes
`username`	User account with RW access to share	Optional
`password_secret`	(Recommended) Secret ID for the password for the user account, which can be used instead of the password.	Optional
`password`	Password for the user account. We recommend that you use the `password_secret` parameter instead.	Optional

> **Note:**
>
> When running the `sr-create` command, we recommend that you use the `device-config:password_secret` argument instead of specifying the password on the command line. For more information, see Secrets.

For example, to create a shared SMB SR on `192.168.1.10:/share1`, use the following command:

```
    xe sr-create content-type=user \
    name-label="Example shared SMB SR" shared=true \
    device-config:server=//192.168.1.10/share1 \
    device-config:username=valid_username device-config:password_secret
       =valid_password_secret type=smb
<!--NeedCopy-->
```

To create a non-shared SMB SR, run the following command:

```
    xe sr-create host-uuid=host_uuid content-type=user \
    name-label="Non-shared SMB SR" \
    device-config:server=//192.168.1.10/share1 \
    device-config:username=valid_username device-config:password_secret
       =valid_password_secret type=smb
<!--NeedCopy-->
```

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 that provides, for example, hardware-based iSCSI or FC support.

XenServer hosts support Fibre Channel SANs through Emulex or QLogic host bus adapters (HBAs). All Fibre Channel configuration required to expose a Fibre Channel LUN to the host must be completed manually. This configuration includes storage devices, network devices, and the HBA within the XenServer host. After all FC configuration is complete, the HBA exposes 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. Therefore, this value must be used when creating shared SRs accessible by all hosts in a resource pool.

The same features apply to QLogic iSCSI HBAs.

See Create 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.
>
> The block size of an LVM over HBA LUN must be 512 bytes. To use storage with 4 KB physical blocks, the storage must also support emulation of 512 byte allocation blocks (the logical block size must be 512 bytes).

Thin-provisioned shared GFS2 block storage

Thin provisioning better utilizes the available storage by allocating disk storage space to VDIs as data is written to the virtual disk, rather than allocating the full virtual size of the VDI in advance. Thin provisioning enables you to significantly reduce the amount of space required on a shared storage array, and with that your Total Cost of Ownership (TCO).

Thin provisioning for shared block storage is of particular interest in the following cases:

- You want increased space efficiency. Images are sparsely and not thickly allocated.
- You want to reduce the number of I/O operations per second on your storage array. The GFS2 SR is the first SR type to support storage read caching on shared block storage.
- You use a common base image for multiple virtual machines. The images of individual VMs will then typically utilize even less space.
- You use snapshots. Each snapshot is an image and each image is now sparse.
- You want to create VDIs that are greater than 2 TiB in size. The GFS2 SR supports VDIs up to 16 TiB in size.
- Your storage doesn't support NFS or SMB3 and only supports block storage. If your storage supports NFS or SMB3, we recommend you use these SR types instead of GFS2.
- Your storage doesn't support thin provisioning of LUNs. If your storage does thin provision LUNs, you can encounter problems and run out of space when combining it with GFS2. Combining GFS2 with a thin-provisioned LUN does not provide many additional benefits and is not recommended.

> **Note:**
>
> We recommend not to use a GFS2 SR with a VLAN due to a known issue where you cannot add or remove hosts on a clustered pool if the cluster network is on a non-management VLAN.

The shared GFS2 type represents disks as a filesystem created on an iSCSI or HBA LUN. VDIs stored on a GFS2 SR are stored in the QCOW2 image format.

This article describes how to set up your GFS2 environment by using the xe CLI. To set up a GFS2 environment by using XenCenter, see the XenCenter product documentation.

1. Plan your GFS2 environment

To provide the benefits of thin provisioning on shared block storage without risk of data loss, your pool must deliver a good level of reliability and connectivity. It is crucial that the hosts in the resource pool that uses GFS2 can reliably communicate with one another. To ensure this, XenServer requires that you use a clustered pool with your GFS2 SR. We also recommend that you design your environment and configure XenServer features to provide as much resiliency and redundancy as possible.

Before setting up your XenServer pool to work with GFS2 SRs, review the following requirements and recommendations for an ideal GFS2 environment:

- **Recommended:** Configure redundant networking infrastructure.
- **Recommended:** Create a dedicated bonded network
- **Required:** Set up a clustered pool
- **Optional** Increase your control domain memory
- **Recommended:** Configure storage multipathing
- **Required:** Create a GFS2 SR

A clustered pool with GFS2 SRs has some differences in behavior to other types of pool and SR. For more information, see Constraints.

2. Configure redundant networking infrastructure

A bonded network links two or more NICs together to create a single channel for network traffic. We recommend that you use a bonded network for your clustered pool traffic. However, before you set up your bonded network, ensure that your network hardware configuration promotes redundancy in the bonded network. Consider implementing as many of these recommendations as is feasible for your organization and environment.

The following best practices add resiliency against software, hardware, or power failures that can affect your network switches.

- Ensure that you have separate physical network switches available for use in the bonded network, not just ports on the same switch.
- Ensure that the separate switches draw power from different, independent power distribution units (PDUs).
- If possible, in your data center, place the PDUs on different phases of the power feed or even feeds provided by different utility companies.
- Consider using uninterruptible power supply units to ensure that the network switches and servers can continue to function or perform an orderly shutdown in the event of a power failure.

3. Create a dedicated bonded network

It is important to ensure that hosts in a clustered pool can communicate reliably with one another. Creating a bonded network for this pool traffic increases the resiliency of your clustered pool.

A bonded network creates a bond between two or more NICs to create a single, high-performing channel that your clustered pool can use for cluster heartbeat traffic. We strongly recommend that this bonded network is not used for any other traffic. Create a separate network for the pool to use for management traffic.

> **Warning:**
>
> If you choose not to follow this recommendation, you are at a higher risk of losing cluster management network packets. Loss of cluster management network packets can cause your clustered pool to lose quorum and some or all hosts in the pool will self-fence.
>
> If your cluster is fencing or facing a problem in this unrecommended configuration, XenServer Support might ask you to reproduce the same problem on a recommended configuration during the course of investigation.

To create a bonded network to use as the clustering network:

1. If you have a firewall between the hosts in your pool, ensure that hosts can communicate on the cluster network using the following ports:

   - TCP: 8892, 8896, 21064
   - UDP: 5404, 5405

   For more information, see Communication ports used by XenServer.

2. Open a console on the XenServer host that you want to act as the pool coordinator.

3. Create a network for use with the bonded NIC by using the following command:

   ```
   xe network-create name-label=bond0
   <!--NeedCopy-->
   ```

   The UUID of the new network is returned.

4. Find the UUIDs of the PIFs to use in the bond by using the following command:

   ```
   xe pif-list
   <!--NeedCopy-->
   ```

5. Create your bonded network in either active-active mode, active-passive mode, or LACP bond mode. Depending on the bond mode you want to use, complete one of the following actions:

   - 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>,<pif_uuid_3>,<
            pif_uuid_4>
     <!--NeedCopy-->
     ```

     Type two UUIDs when you are bonding two NICs and four UUIDs when you are bonding four NICs. The UUID for the bond is returned after running the command.

- To configure the bond in active-passive or LACP bond mode, use the same syntax, add the optional `mode` parameter, and specify `lacp` or `active-backup`:

  ```
  xe bond-create network-uuid=<network_uuid> /
      pif-uuids=<pif_uuid_1>,<pif_uuid_2>,<pif_uuid_3>,<
          pif_uuid_4> /
      mode=balance-slb | active-backup | lacp
  <!--NeedCopy-->
  ```

After you create your bonded network on the pool coordinator, when you join other XenServer hosts to the pool, the network and bond information is automatically replicated to the joining server.

For more information, see Networking.

> **Note:**
>
> - Changing the IP address of the cluster network by using XenCenter requires clustering and GFS2 to be temporarily disabled.
> - Do not change the bonding of your clustering network while the cluster is live and has running VMs. This action can cause hosts in the cluster to hard restart (fence).
> - If you have an IP address conflict (multiple hosts having the same IP address) on your clustering network involving at least one host with clustering enabled, the cluster does not form correctly and the hosts are unable to fence when required. To fix this issue, resolve the IP address conflict.

To test your active-passive bonded network failover times:

For bonded networks that use active-passive mode, if the active link fails, there is a failover period when the network link is broken while the passive link becomes active. If the time it takes for your active-passive bonded network to fail over is longer than the cluster timeout, some or all hosts in your clustered pool might still fence.

You can test your bonded network failover time by forcing the network to fail over by using one of the following methods:

- By physically pulling out the network cables
- By disabling switch ports on one network link

Repeat the test a number of times to ensure the result is consistent.

The cluster timeout value of your pool depends on how many hosts are in your cluster. Run the following command to find the `token-timeout` value in seconds for the pool:

```
xe cluster-param-get uuid=<cluster_uuid> param-name=token-timeout
```

If the failover time is likely to be greater than the timeout value, your network infrastructure and configuration might not be reliable enough to support a clustered pool.

4. Set up a clustered pool

To use shared GFS2 storage, the XenServer resource pool must be a clustered pool. Enable clustering on your pool before creating a GFS2 SR.

A clustered pool is a pool of XenServer hosts that are more closely connected and coordinated than hosts in non-clustered pools. The hosts in the cluster maintain constant communication with each other on a selected network. All hosts in the cluster are aware of the state of every host in the cluster. This host coordination enables the cluster to control access to the contents of the GFS2 SR. To ensure that the clustered pool always remains in communication, each host in a cluster must always be in communication with at least half of the hosts in the cluster (including itself). This state is known as a host having quorum. If a host does not have quorum, it hard restarts and removes itself from the cluster. This action is referred to as 'fencing'.

For more information, see Clustered pools.

Before you start setting up your clustered pool, ensure that the following prerequisites are met:

- Plan to create a pool of between 3 and 16 hosts.

  Where possible, use an odd number of hosts in a clustered pool as this ensures that hosts are always able to determine if they have quorun. We recommend that you use clustering only in pools containing at least three hosts, as pools of two hosts are sensitive to self-fencing the entire pool.

  Clustered pools only support up to 16 hosts per pool.
- All XenServer hosts in the clustered pool must have at least 2 GiB of control domain memory.
- All hosts in the cluster must use static IP addresses for the cluster network.
- If you are clustering an existing pool, ensure that high availability is disabled. You can enable high availability again after clustering is enabled.

To use the xe CLI to create a clustered pool:

1. Create a resource pool of at least three XenServer hosts.

   Repeat the following steps on each joining XenServer host that is not the pool coordinator:

   a) Open a console on the XenServer host.

   b) Join the XenServer host to the pool on the pool coordinator by using the following command:

   ```
   xe pool-join master-address=<master_address> /
       master-username=<administrators_username> /
       master-password=<password>
   <!--NeedCopy-->
   ```

The value of the `master-address` parameter must be set to the fully qualified domain name of the XenServer host that is the pool coordinator. The `password` must be the administrator password set when the pool coordinator was installed.

For more information, see Hosts and resource pools.

2. For every PIF that belongs to this network, set `disallow-unplug=true`.

   a) Find the UUIDs of the PIFs that belong to the network by using the following command:

   ```
   xe pif-list
   <!--NeedCopy-->
   ```

   b) Run the following command on a XenServer host in your resource pool:

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

3. Enable clustering on your pool. Run the following command on a XenServer host in your resource pool:

   ```
   xe cluster-pool-create network-uuid=<network_uuid>
   <!--NeedCopy-->
   ```

   Provide the UUID of the bonded network that you created in an earlier step.

5. Increase your control domain memory

If you have insufficient control domain memory on your hosts, your pool can experience network instabililty. Network instability can cause problems for a clustered pool with GFS2 SRs.

It is important to ensure that your clustered pool has an appropriate amount of control domain memory. For information about changing the amount of control domain memory and monitoring the memory behavior, see Memory usage.

6. Configure storage multipathing

Ensure that storage multipathing is set up between your clustered pool and your GFS2 SR.

Multipathing routes storage traffic to a storage device over multiple paths for redundancy. All routes can have active traffic on them during normal operation, which results in increased throughput.

Before enabling multipathing, verify that the following statements are true:

- Your ethernet or fibre switch is configured to make multiple targets available on your storage server.

For example, an iSCSI storage back-end queried for `sendtargets` on a given portal returns 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
```

However, you can perform additional configuration to enable iSCSI multipath for arrays that only expose a single target. For more information, see iSCSI multipath for arrays that only expose a single target.

- For iSCSI only, the control domain (dom0) has an IP address on each subnet used by the multipathed storage.

  Ensure that for each path to the storage, you have a NIC and that there is an IP address configured on each NIC. For example, if you want four paths to your storage, you must have four NICs that each have an IP address configured.

- For iSCSI only, every iSCSI target and initiator has a unique IQN.
- For iSCSI only, the iSCSI target ports are operating in portal mode.
- For HBA only, multiple HBAs are connected to the switch fabric.
- If possible, use multiple redundant switches.

To enable multipathing by using the xe CLI

We recommend that you enable multipathing for all hosts in your pool *before* creating the SR. If you create the SR before enabling multipathing, you must put your hosts into maintenance mode to enable multipathing.

1. Open a console on the XenServer host.
2. Unplug all PBDs on the host by using the following command:

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

   You can use the command `xe pbd-list` to find the UUID of the PBDs.

3. Set the value of the `multipathing` parameter to **`true`** by using the following command:

   ```
   xe host-param-set uuid=<host uuid> multipathing=true
   <!--NeedCopy-->
   ```

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

   - Migrate or suspend any running guests with virtual disks in the affected SRs.

- Replug the PBD of any affected SRs to reconnect them using multipathing:

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

5. Repeat these steps to enable multipathing on all hosts in the pool.

Ensure that you enable multipathing on all hosts in the pool. All cabling and, in the case of iSCSI, subnet configurations must match the corresponding NICs on each host.

For more information, see Storage multipathing.

7. Create a GFS2 SR

Create your shared GFS2 SR on an iSCSI or an HBA LUN that is visible to all XenServer hosts in your resource pool. We do not recommend using a thin-provisioned LUN with GFS2. However, if you do choose this configuration, you must ensure that the LUN always has enough space to allow XenServer to write to it.

You can add up to 62 GFS2 SRs to a clustered pool.

If you have previously used your block-based storage device for thick provisioning with LVM, this is detected by XenServer. XenCenter gives you the opportunity to use the existing LVM partition or to format the disk and set up a GFS2 partition.

Create a shared GFS2 over iSCSI SR

You can create GFS2 over iSCSI SRs by using XenCenter. For more information, see Software iSCSI storage in the XenCenter product documentation.

Alternatively, you can use the xe CLI to create a GFS2 over iSCSI SR.

Device-config parameters for GFS2 SRs:

Parameter Name	Description	Required?
`provider`	The block provider implementation. In this case, `iscsi`.	Yes
`target`	The IP address or hostname of the iSCSI filer that hosts	Yes
`targetIQN`	The IQN target of iSCSI filer that hosts the SR	Yes
`SCSIid`	Device SCSI ID	Yes

You can find the values to use for these parameters by using the `xe sr-probe-ext` command.

```
xe sr-probe-ext type=<type> host-uuid=<host_uuid> device-config:=<config> sm-config:=<sm_config>
<!--NeedCopy-->
```

1. Start by running the following command:

   ```
   xe sr-probe-ext type=gfs2 device-config:provider=iscsi
   <!--NeedCopy-->
   ```

   The output from the command prompts you to supply additional parameters and gives a list of possible values at each step.

2. Repeat the command, adding new parameters each time.

3. When the command output starts with `Found the following complete configurations that can be used to create SRs:`, you can locate the SR by using the `xe sr-create` command and the `device-config` parameters that you specified.

   Example output:

   ```
   Found the following complete configurations that can be used to create SRs:
   Configuration 0:
     SCSIid       : 36001405852f77532a064687aea8a5b3f
         targetIQN: iqn.2009-01.example.com:iscsi192a25d6
            target: 198.51.100.27
          provider: iscsi


   Configuration 0 extra information:
   <!--NeedCopy-->
   ```

To create a shared GFS2 SR on a specific LUN of an iSCSI target, run the following command on a server in your clustered pool:

```
xe sr-create type=gfs2 name-label="Example GFS2 SR" --shared \
   device-config:provider=iscsi device-config:targetIQN=<target_iqns> \
   device-config:target=<portal_address> device-config:SCSIid=<scsci_id>
<!--NeedCopy-->
```

If the iSCSI target is not reachable while GFS2 filesystems are mounted, some hosts in the clustered pool might hard restart (fence).

For more information about working with iSCSI SRs, see Software iSCSI support.

Create a shared GFS2 over HBA SR

You can create GFS2 over HBA SRs by using XenCenter. For more information, see Hardware HBA storage in the XenCenter product documentation.

Alternatively, you can use the xe CLI to create a GFS2 over HBA SR.

Device-config parameters for GFS2 SRs:

Parameter name	Description	Required?
`provider`	The block provider implementation. In this case, `hba`.	Yes
`SCSIid`	Device SCSI ID	Yes

You can find the values to use for the SCSIid parameter by using the `xe sr-probe-ext` command.

```
xe sr-probe-ext type=<type> host-uuid=<host_uuid> device-config:=<
    config> sm-config:=<sm_config>
<!--NeedCopy-->
```

1. Start by running the following command:

   ```
   xe sr-probe-ext type=gfs2 device-config:provider=hba
   <!--NeedCopy-->
   ```

   The output from the command prompts you to supply additional parameters and gives a list of possible values at each step.

2. Repeat the command, adding new parameters each time.

3. When the command output starts with `Found the following complete configurations that can be used to create SRs:`, you can locate the SR by using the `xe sr-create` command and the `device-config` parameters that you specified.

   Example output:

   ```
   Found the following complete configurations that can be used to
       create SRs:
   Configuration 0:
     SCSIid        : 36001405852f77532a064687aea8a5b3f
         targetIQN: iqn.2009-01.example.com:iscsi192a25d6
            target: 198.51.100.27
          provider: iscsi


   ```

```
Configuration 0 extra information:
<!--NeedCopy-->
```

To create a shared GFS2 SR on a specific LUN of an HBA target, run the following command on a server in your clustered pool:

```
xe sr-create type=gfs2 name-label="Example GFS2 SR" --shared \
  device-config:provider=hba device-config:SCSIid=<device_scsi_id>
<!--NeedCopy-->
```

For more information about working with HBA SRs, see Hardware host bus adapters.

What's next?

Now that you have your GFS2 environment set up, it is important that you maintain the stability of your clustered pool by ensuring it has quorum. For more information, see Manage your clustered pool.

If you encounter issues with your GFS2 environment, see Troubleshoot clustered pools.

You can manage your GFS2 SR the same way as you do other SRs. For example, you can add capacity to the storage array to increase the size of the LUN. For more information, see Live LUN expansion.

Constraints

Shared GFS2 storage currently has the following constraints:

- As with any thin-provisioned SR, if the GFS2 SR usage grows to 100%, further writes from VMs fail. These failed writes can then lead to failures within the VM, possible data corruption, or both.
- XenCenter shows an alert when your SR usage grows to 80%. Ensure that you monitor your GFS2 SR for this alert and take the appropriate action if seen. On a GFS2 SR, high usage causes a performance degradation. We recommend that you keep your SR usage below 80%.
- VM migration with storage migration (live or offline) is not supported for VMs whose VDIs are on a GFS2 SR. You also cannot migrate VDIs from another type of SR to a GFS2 SR.
- The Software FCoE transport is not supported with GFS2 SRs (for fully offloaded FCoE use HBA).
- Trim/unmap is not supported on GFS2 SRs.
- CHAP is not supported on GFS2 SRs.
- Performance metrics are not available for GFS2 SRs and disks on these SRs.
- Changed block tracking is not supported for VDIs stored on GFS2 SRs.

- You cannot export VDIs that are greater than 2 TiB as VHD or OVA/OVF. However, you can export VMs with VDIs larger than 2 TiB in XVA format.
- We do not recommend using a thin-provisioned LUN with GFS2. However, if you do choose this configuration, you must ensure that the LUN always has enough space to allow XenServer to write to it.
- We do not recommend using SAN deduplication with GFS2 SRs. However, if you do choose this configuration, you must use suitable external monitoring of your SAN utilization to ensure that there is always space for XenServer to write to.
- Your GFS2 file system cannot be larger than 100 TiB.
- You cannot have more than 62 GFS2 SRs in your pool.
- Clustered pools only support up to 16 hosts per pool.
- To enable HA on your clustered pool, the heartbeat SR must be a GFS2 SR.
- For cluster traffic, we strongly recommend that you use a bonded network that uses at least two different network switches. Do not use this network for any other purposes.
- Changing the IP address of the cluster network by using XenCenter requires clustering and GFS2 to be temporarily disabled.
- Do not change the bonding of your clustering network while the cluster is live and has running VMs. This action can cause hosts in the cluster to hard restart (fence).
- If you have an IP address conflict (multiple hosts having the same IP address) on your clustering network involving at least one host with clustering enabled, the cluster does not form correctly and the hosts are unable to fence when required. To fix this issue, resolve the IP address conflict.

Manage storage repositories

April 8, 2024

This section covers creating storage repository types and making them available to your XenServer host. It also covers various operations required in the ongoing management of Storage Repositories (SRs), including Live VDI Migration.

Create storage repositories

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

> **Note:**
>
> Local SRs of type `lvm`, `ext`, and `xfs` can only be created using the xe CLI. After creation, you can manage all SR types by either XenCenter or the xe CLI.

There are two basic steps to create a storage repository for use on a host by 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 be *destroyed* when no longer in use to free up the physical device. SRs can also be *forgotten* to detach the SR from one XenServer host and attach it to another. For more information, see *Removing SRs* in the following section.

Probe an SR

The `sr-probe` command can be used in the following ways:

- To identify unknown parameters for use in creating an SR
- 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. If 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. The set of IQNs available on the target is 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>
<!--NeedCopy-->
```

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>
<!--NeedCopy-->
```

Probing the same target and supplying all three parameters returns 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=149455400000000000000000002000000
       b70200000f000000

    <?xml version="1.0" ?>
    <SRlist>
        <SR>
            <UUID>
                3f6e1ebd-8687-0315-f9d3-b02ab3adc4a6
            </UUID>
```

```
                <Devlist>
                    /dev/disk/by-id/scsi-149455400000000000000000002000000
                        b70200000f000000
                </Devlist>
            </SR>
        </SRlist>
<!--NeedCopy-->
```

The following parameters can be probed for each SR type:

SR type	The `device-config` parameters, 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
`lvmofcoe`	`SCSIid`	Yes	Yes
`nfs`	`server`	No	Yes
	`serverpath`	Yes	Yes
`smb`	`server`	No	Yes
	`username`	No	No
	`password`	No	No
`lvm`	`device`	No	Yes
`ext`	`device`	No	Yes

For information about probing a GFS2 SR, see Create a GFS2 SR.

Remove SRs

A Storage Repository (SR) can be removed either temporarily or permanently.

Detach: Breaks the association between the storage device and the pool or host (PBD Unplug). The SR (and its VDIs) becomes inaccessible. The contents of the VDIs and the meta-information used by VMs to access the VDIs are preserved. Detach can be used when you temporarily take an SR offline, for example, for maintenance. A detached SR can later be reattached.

Forget: Preserves the contents of the SR on the physical disk, but the information that connects a VM to its VDIs is permanently deleted. For example, allows you to reattach the SR, to another XenServer host, without removing any of the SR contents.

Destroy: Deletes the contents of the SR from the physical disk.

> **Note:**
>
> When using SMB storage, do not remove the share from the storage before detaching the SMB SR.

For Destroy or Forget, the PBD connected to the SR must be unplugged from the host.

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

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

2. Use the `sr-destroy` command to remove an SR. The command destroys the SR, deletes the SR and corresponding PBD from the XenServer host database and deletes the SR contents from the physical disk:

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

3. Use the `sr-forget` command to forget an SR. The command removes the SR and corresponding PBD from the XenServer host database but leaves the actual SR content intact on the physical media:

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

> **Note:**
>
> It can take some time for the software object corresponding to the SR to be garbage collected.

Introduce an SR

To reintroduce a previously *forgotten* SR, create a PBD. Manually plug the PBD to the appropriate XenServer hosts to activate the SR.

The following example introduces an 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=149455400000000000000000002000000
           b70200000f000000
   ```

```
<!--NeedCopy-->
```

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
   <!--NeedCopy-->
   ```

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=149455400000000000000000002000000
           b70200000f000000
   <!--NeedCopy-->
   ```

4. Plug the PBD to attach the SR:

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

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

   ```
   xe pbd-list sr-uuid=sr_uuid
   <!--NeedCopy-->
   ```

> **Note:**
>
> Perform steps 3 through 5 for each host in the resource pool. These steps can also be performed using the Repair Storage Repository function in XenCenter.

Live LUN expansion

To fulfill capacity requirements, you may need to add capacity to the storage array to increase the size of the LUN provisioned to the XenServer host. Live LUN Expansion allows to you to increase the size of the LUN without any VM downtime.

After adding more capacity to your storage array, enter,

```
xe sr-scan sr-uuid=sr_uuid
<!--NeedCopy-->
```

This command rescans the SR, and any extra capacity is added and made available.

This operation is also available in XenCenter. Select the SR to resize, and then click **Rescan**.

> **Warnings:**
>
> - It is not possible to shrink or truncate LUNs. Reducing the LUN size on the storage array can lead to data loss.

Live VDI migration

Live VDI migration allows the administrator to relocate the VMs Virtual Disk Image (VDI) without shutting down the VM. This feature enables administrative operations such as:

- Moving a VM from cheap local storage to fast, resilient, array-backed storage.
- Moving a VM from a development to production environment.
- Moving between tiers of storage when a VM is limited by storage capacity.
- Performing storage array upgrades.

Limitations and caveats

Live VDI Migration is subject to the following limitations and caveats

- There must be sufficient disk space available on the target repository.

To move virtual disks by using XenCenter

1. In the **Resources** pane, select the SR where the Virtual Disk is stored and then click the **Storage** tab.
2. In the **Virtual Disks** list, select the Virtual Disk that you would like to move, and then click **Move**.
3. In the **Move Virtual Disk** dialog box, select the target SR that you would like to move the VDI to.

   > **Note:**
   >
   > Ensure that the SR has sufficient space for another virtual disk: the available space is shown in the list of available SRs.

4. Click **Move** to move the virtual disk.

For xe CLI reference, see `vdi-pool-migrate`.

Cold VDI migration between SRs (offline migration)

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

For xe CLI reference, see `vm-migrate`.

Copy all of a VM's VDIs to a different 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. Shut down the VM.
2. Within XenCenter, select the VM and then select the **VM** > **Copy VM** option.
3. Select the desired target SR.

Copy individual VDIs to a different SR

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

1. Shut down 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` is listed as `not in database` and can be ignored.

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

   > **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, first check if the VBD has the parameter `other-config:owner` set to `true`. Set this parameter to `false`. Issuing the `vbd-destroy` command with `other-config:owner=true` also destroys 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
   <!--NeedCopy-->
   ```

5. In 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, select the **Storage** tab of the original SR in XenCenter. The original VDIs are listed with an empty value for the VM field. Use the **Delete** button to delete the VDI.

Convert 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 8.
2. Ensure that all hosts in the pool have the SR's LUN zoned appropriately. See Probe an SR for details on using the `sr-probe` command to verify that the LUN is present on each host.
3. Convert the SR to shared:

   ```
   xe sr-param-set shared=true uuid=local_fc_sr
   <!--NeedCopy-->
   ```

4. The SR is moved from the host level to the pool level in XenCenter, indicating that it is now shared. The SR is 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** option.
6. Click **Repair** to create and plug a PBD for each host in the pool.

Reclaim space for block-based storage on the backing array using discard

You can use space reclamation to free up unused blocks on a thinly provisioned LUN. After the space is released, the storage array can then reuse this reclaimed space.

> **Note:**
>
> Space reclamation is only available on some types of storage arrays. To determine whether your array supports this feature and whether it needs a specific configuration, see the Hardware Compatibility List and your storage vendor specific documentation.

To reclaim the space by using XenCenter:

1. Select the **Infrastructure** view, and then choose the host or pool connected to the SR.
2. Click the **Storage** tab.
3. Select the SR from the list, and click **Reclaim freed space**.

4. Click **Yes** to confirm the operation.
5. Click **Notifications** and then **Events** to view the status of the operation.

For more information, press `F1`in XenCenter to access the Online Help.

To reclaim space by using the xe CLI, you can use the following command:

```
xe host-call-plugin host-uuid=host_uuid \
    plugin=trim fn=do_trim args:sr_uuid=sr_uuid
```

> **Notes:**
>
> - The operation is only available for LVM-based SRs that are based on thinly provisioned LUNs on the array. Local SSDs can also benefit from space reclamation.
> - Space reclamation is not required for file-based SRs such as NFS and EXT3/EXT4. The **Reclaim Freed Space** button is not available in XenCenter for these SR types.
> - If you run the space reclamation xe command for a file-based SR or a thick-provisioned LVM-based SR, the command returns an error.
> - Space reclamation is an intensive operation and can lead to a degradation in storage array performance. Therefore, only initiate this operation when space reclamation is required on the array. We recommend that you schedule this work outside of peak array demand hours.

Automatically reclaim space when deleting snapshots

When deleting snapshots with XenServer, space allocated on LVM-based SRs is reclaimed automatically and a VM reboot is not required. This operation is known as 'online coalescing'. Online coalescing applies to all types of SR.

In certain cases, automated space reclamation might be unable to proceed. We recommend that you use the offline coalesce tool in these scenarios:

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

> **Notes:**
>
> - Running the offline coalesce tool incurs 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 tool reclaims as much space as possible given the remaining snapshots/clones. If you want to reclaim the entire space, delete all snapshots and clones.
> - 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.

Reclaim space by using the offline coalesce tool

Enable the hidden objects using XenCenter. Click **View** > **Hidden** objects. In the Resource pane, select the VM for which you want to obtain the UUID. The UUID is displayed in the **General** tab.

In the **Resource** pane, select the resource pool coordinator (the first host in the list). The **General** tab displays the UUID. If you are not using a resource pool, select the VM's 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
   <!--NeedCopy-->
   ```

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

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

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

Notes:

We recommend that you shut down or suspend the VM manually before running the off-line coalesce tool. You can shut down or suspend the VM using either XenCenter or the XenServer CLI. If you run the coalesce tool on a running VM, the tool automatically suspends the VM, performs the required VDI coalesce operations, and resumes the VM. Agile VMs might restart on a different host.

If the Virtual Disk Images (VDIs) to be coalesced are on shared storage, you must run the off-line coalesce tool on the pool coordinator.

If the VDIs to be coalesced are on local storage, run the off-line coalesce tool on the host to which the local storage is attached.

Working with disk I/O

You can configure the disk I/O scheduler and the disk I/O priority settings to change the performance of your disks.

> **Note:**
>
> The disk I/O capabilities described in this section do not apply to EqualLogic, NetApp, or NFS storage.

Adjust the disk I/O 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.

1. Adjust the disk scheduler by using the following command:

   ```
   xe sr-param-set other-config:scheduler=<option> uuid=<sr_uuid>
   <!--NeedCopy-->
   ```

   The value of `<option>` can be one of the following terms: `noop`, `cfq`, or `deadline`.

2. Unplug and replug the corresponding PBD for the scheduler parameter to take effect.

   ```
   xe pbd-unplug uuid=<pbd_uuid>
   xe pbd-plug uuid=<pbd_uuid>
   <!--NeedCopy-->
   ```

To apply disk I/O request prioritization, override the default setting and assign the `cfq` disk scheduler to the SR.

Virtual disk I/O request prioritization

Virtual disks have optional I/O request priority settings. You can use these settings to prioritize I/O to a particular VM's disk over others.

Before configuring any disk I/O request priority parameters for a VBD, ensure that the disk scheduler for the SR has been set appropriately. The scheduler parameter must be set to `cfq` on the SR and the associated PBD unplugged and replugged. For information about how to adjust the scheduler, see Adjusting the disk I/O scheduler.

For shared SR, where multiple hosts are accessing the same LUN, the priority setting is applied to VBDs accessing the LUN from the same host. These settings are not applied across hosts in the pool.

The host issues a request to the remote storage, but the request prioritization is done by the remote storage.

Setting disk I/O request parameters These settings can be applied to existing virtual disks by using the `xe vbd-param-set` command with the following parameters:

- `qos_algorithm_type` - This parameter must be set to the value `ionice`, which is the only algorithm supported for virtual disks.
- `qos_algorithm_param` - Use this parameter to set key/value pairs. For virtual disks, `qos_algorithm_param` takes a `sched` key, and depending on the value, also requires a `class` key.

  The key `qos_algorithm_param:sched` can have one of the following values:

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

  The key `qos_algorithm_param:class` can have one of the following values:

  - 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. For example, I/O requests with a priority of 5, are given priority over I/O requests with a priority of 2.

Example 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-unplug uuid=<pbd_uuid>
xe pbd-plug uuid=<pbd_uuid>
<!--NeedCopy-->
```

Storage multipathing

April 29, 2024

Dynamic multipathing support is available for Fibre Channel and iSCSI storage back-ends.

XenServer uses Linux native multipathing (DM-MP), the generic Linux multipathing solution, as its multipath handler. However, XenServer supplements this handler with additional features so that XenServer can recognize vendor-specific features of storage devices.

Configuring multipathing provides redundancy for remote storage traffic if there is partial connectivity loss. Multipathing routes storage traffic to a storage device over multiple paths for redundancy and increased throughput. You can use up to 16 paths to a single LUN. Multipathing is an active-active configuration. By default, multipathing uses either round-robin or multibus load balancing depending on the storage array type. All routes have active traffic on them during normal operation, which results in increased throughput.

> **Important:**
>
> We recommend that you enable multipathing for all hosts in your pool *before* creating the SR. If you create the SR before enabling multipathing, you must put your hosts into maintenance mode to enable multipathing.

NIC bonding can also provide redundancy for storage traffic. For iSCSI storage, we recommend configuring multipathing instead of NIC bonding whenever possible.

Multipathing is not effective in the following scenarios:

- NFS storage devices
- You have limited number of NICs and need to route iSCSI traffic and file traffic (NFS or SMB) over the same NIC

In these cases, consider using NIC bonding instead. For more information about NIC bonding, see Networking.

Prerequisites

Before enabling multipathing, verify that the following statements are true:

- Multiple targets are available on your storage server.

  For example, an iSCSI storage back-end queried for `sendtargets` on a given portal returns multiple targets, as in the following example:

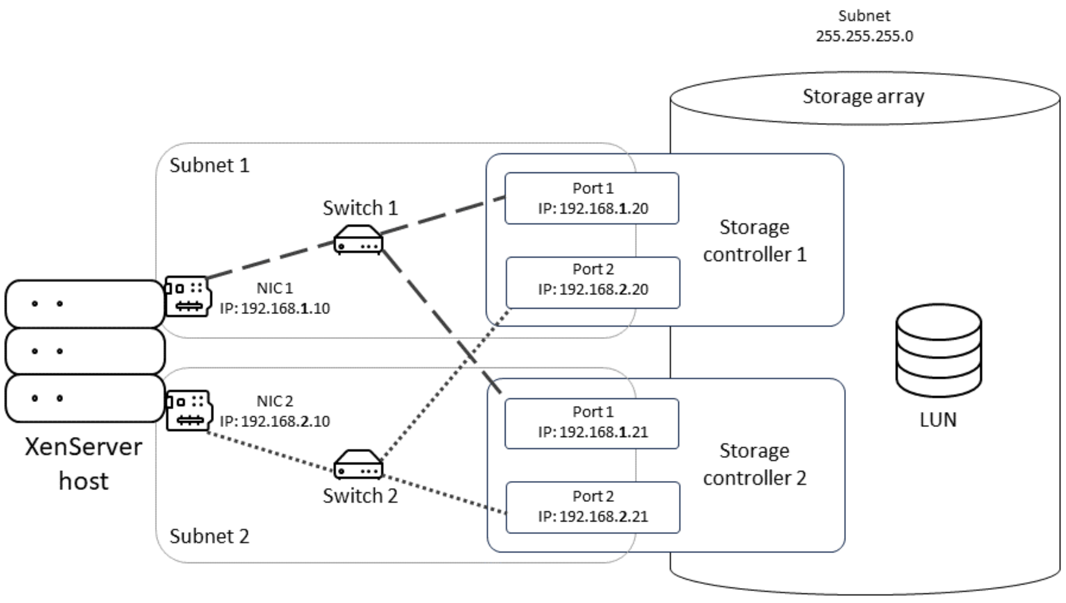
  ```
  iscsiadm -m discovery --type sendtargets --portal 192.168.0.161
  192.168.0.161:3260,1 iqn.strawberry:litchie
  192.168.0.204:3260,2 iqn.strawberry:litchie
  ```

  However, you can perform additional configuration to enable iSCSI multipath for arrays that only expose a single target. For more information, see iSCSI multipath for arrays that only expose a single target.

- For iSCSI only, the control domain (dom0) has an IP address on each subnet used by the multipathed storage.

Ensure that for each path you want to have to the storage, you have a NIC and that there is an IP address configured on each NIC. For example, if you want four paths to your storage, you must have four NICs that each have an IP address configured.

- For iSCSI only, every iSCSI target and initiator has a unique IQN.
- For iSCSI only, the iSCSI target ports are operating in portal mode.
- For HBA only, multiple HBAs are connected to the switch fabric.
- When you are configuring secondary interfaces, each secondary interface must be on a separate subnet. For example, if you want to configure two more secondary interfaces for storage, you require IP addresses on three different subnets –one subnet for the management interface, one subnet for Secondary Interface 1, and one subnet for Secondary Interface 2.



This diagram shows how both NICs on the host in a multipathed iSCSI configuration must be on different subnets. In this diagram, NIC 1 on the host along with Switch 1 and NIC 1 on both storage controllers are on a different subnet than NIC2, Switch 2, and NIC 2 on the storage controllers.

Enable multipathing

You can enable multipathing in XenCenter or on the xe CLI.

To enable multipathing by using XenCenter

1. In the XenCenter **Resources** pane, right-click on the host and choose **Enter Maintenance Mode**.

2. Wait until the host reappears in the **Resources** pane with the maintenance mode icon (a blue square) before continuing.
3. On the **General** tab for the host, click **Properties** and then go to the **Multipathing** tab.
4. To enable multipathing, select the **Enable multipathing on this server** check box.
5. Click **OK** to apply the new setting. There is a short delay while XenCenter saves the new storage configuration.
6. In the **Resources** pane, right-click on the host and choose **Exit Maintenance Mode**.
7. Repeat these steps to enable multipathing on all hosts in the pool.

Ensure that you enable multipathing on all hosts in the pool. All cabling and, in the case of iSCSI, subnet configurations must match the corresponding NICs on each host.

To enable multipathing by using the xe CLI

1. Open a console on the XenServer host.
2. Unplug all PBDs on the host by using the following command:

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

   You can use the command `xe pbd-list` to find the UUID of the PBDs.
3. Set the value of the `multipathing` parameter to `true` by using the following command:

   ```
   xe host-param-set uuid=<host uuid> multipathing=true
   <!--NeedCopy-->
   ```

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

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

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

5. Repeat these steps to enable multipathing on all hosts in the pool.

Ensure that you enable multipathing on all hosts in the pool. All cabling and, in the case of iSCSI, subnet configurations must match the corresponding NICs on each host.

Disable multipathing

You can disable multipathing in XenCenter or on the xe CLI.

To disable multipathing by using XenCenter

1. In the XenCenter **Resources** pane, right-click on the host and choose **Enter Maintenance Mode**.
2. Wait until the host reappears in the **Resources** pane with the maintenance mode icon (a blue square) before continuing.
3. On the **General** tab for the host, click **Properties** and then go to the **Multipathing** tab.
4. To disable multipathing, clear the **Enable multipathing on this server** check box.
5. Click **OK** to apply the new setting. There is a short delay while XenCenter saves the new storage configuration.
6. In the **Resources** pane, right-click on the host and choose **Exit Maintenance Mode**.
7. Repeat these steps to configure multipathing on all hosts in the pool.

To disable multipathing by using the xe CLI

1. Open a console on the XenServer host.
2. Unplug all PBDs on the host by using the following command:

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

   You can use the command `xe pbd-list` to find the UUID of the PBDs.
3. Set the value of the `multipathing` parameter to `false` by using the following command:

   ```
   xe host-param-set uuid=<host uuid> multipathing=false
   <!--NeedCopy-->
   ```

4. If there are existing SRs on the server 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 replug the PBD of any affected SRs to reconnect them using multipathing:

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

5. Repeat these steps to disable multipathing on all hosts in the pool.

Configure multipathing

To do additional multipath configuration, create files with the suffix `.conf` in the directory `/etc/multipath/conf.d`. Add the additional configuration in these files. Multipath searches the directory alphabetically for files ending in `.conf` and reads configuration information from them.

Do not edit the file /etc/multipath.conf. This file is overwritten by updates to XenServer.

Multipath tools

Multipath support in XenServer is based on the device-mapper multipathd components. The Storage Manager API handles activating and deactivating multipath nodes automatically. Unlike the standard dm-multipath tools in Linux, device mapper nodes are not automatically created for all LUNs on the system. Device mapper nodes are only provisioned when LUNs are actively used by the storage management layer. Therefore, it is unnecessary to use any of the dm-multipath CLI tools to query or refresh DM table nodes in XenServer.

If it is 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
<!--NeedCopy-->
```

```
mpathutil status
<!--NeedCopy-->
```

iSCSI multipathing on a single subnet

You can configure XenServer to use iSCSI multipath with storage arrays that expose their targets and IQN(s) on a single subnet. For example, you can follow these steps to set up Dell EqualLogic PS and FS unified series storage arrays.

By default, XenServer establishes only one connection per iSCSI target. Hence, with the default configuration the recommendation is to use NIC bonding to achieve failover and load balancing. The configuration procedure outlined in this section describes an alternative configuration, where multiple iSCSI connections are established for a single subnet or target. NIC bonding is not required.

> **Note:**
>
> The following configuration is only supported for servers that are exclusively attached to storage arrays which expose their targets and IQN(s) through a single subnet. These storage arrays must be qualified for this procedure with XenServer.

To configure multipath:

1. Back up any data you want to protect.
2. In the XenCenter **Resources** pane, right-click on the host and choose **Enter Maintenance Mode**.
3. Wait until the host reappears in the **Resources** pane with the maintenance mode icon (a blue square) before continuing.

4. On the **General** tab for the host, click **Properties** and then go to the **Multipathing** tab.
5. To enable multipathing, select the **Enable multipathing on this server** check box.
6. Click **OK** to apply the new setting. There is a short delay while XenCenter saves the new storage configuration.
7. In the host console, configure two to four Open-iSCSI interfaces. Each iSCSI interface is used to establish a separate path. The following steps show the process for two interfaces:

    a) Configure two iSCSI interfaces, run the following commands:

    ```
    iscsiadm -m iface --op new -I c_iface1
    iscsiadm -m iface --op new -I c_iface2
    ```

    Ensure that the interface names have the prefix `c_`. If the interfaces do not use this naming standard, they are ignored and instead the default interface is used.

    > **Note:**
    >
    > This configuration leads to the default interface being used for all connections. This indicates that all connections are being established using a single interface.

    b) Bind the iSCSI interfaces to xenbr1 and xenbr2, by using the following commands:

    ```
    iscsiadm -m iface --op update -I c_iface1 -n iface.
        net_ifacename -v xenbr1
    iscsiadm -m iface --op update -I c_iface2 -n iface.
        net_ifacename -v xenbr2
    ```

    > **Note:**
    >
    > This configuration assumes that the network interfaces configured for the control domain (including xenbr1 and xenbr2) and xenbr0 are used for management. It also assumes that the NIC cards being used for the storage network are NIC1 and NIC2. If this is not the case, refer to your network topology to discover the network interfaces and NIC cards to use in these commands.

8. In the XenCenter **Resources** pane, right-click on the host and choose **Exit Maintenance Mode**. Do not resume your VMs yet.
9. In the host console, run the following commands to discover and log in to the sessions:

    ```
    iscsiadm -m discovery -t st -p <IP of SAN>
    iscsiadm -m node -L all
    ```

10. Delete the stale entries containing old session information by using the following commands:

    ```
    cd /var/lib/iscsi/send_targets/<IP of SAN and port, use ls command
        to check that>
    ```

```
rm -rf <iqn of SAN target for that particular LUN>

cd /var/lib/iscsi/nodes/
rm -rf <entries for that particular SAN>
```

11. Detach the LUN and attach it again. You can do this in one of the following ways:

    - After completing the preceding steps on all hosts in a pool, you can use XenCenter to detach and reattach the LUN for the entire pool.
    - Alternatively, you can unplug and destroy the PBD for each host and then repair the SR.

      a) Run the following commands to unplug and destroy the PBD:

         i. Find the UUID of the SR:

         ```
         xe sr-list
         ```

         ii. Get the list of PBDs associated with the SR:

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

         iii. In the output of the previous command, look for the UUID of the PBD of the iSCSI Storage Repository with a mismatched SCSI ID.

         iv. Unplug and destroy the PBD you identified.

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

      b) Repair the storage in XenCenter.

12. You can now resume your VMs.

Storage read caching

April 10, 2024

Read caching improves a VM's disk performance as, after the initial read from external disk, data is cached within the host's free memory. It improves performance in situations where many VMs are cloned off a single base VM, as it drastically reduces the number of blocks read from disk. For example, in Citrix Virtual Desktops environment Machine Creation Services (MCS) environments.

The performance improvement can be seen whenever data is read from disk more than once, as it gets cached in memory. This change is most noticeable in the degradation of service that occurs during heavy I/O situations. For example, in the following situations:

- When a significant number of end users boot up within a very narrow time frame (boot storm)
- When a significant number of VMs are scheduled to run malware scans at the same time (antivirus storms).

Read caching is enabled by default when you have the appropriate license type.

> **Note:**
>
> Storage Read Caching is available for XenServer Premium Edition customers.

Enable and disable read caching

For file-based SRs, such as NFS, EXT3/EXT4, SMB, and GFS2 SR types, read-caching is enabled by default. Read-caching is disabled for all other SRs.

To disable read caching for a specific SR by using the xe CLI, run the following command:

```
xe sr-param-set uuid=sr-uuid other-config:o_direct=true
<!--NeedCopy-->
```

To disable read caching for a specific SR by using XenCenter, go to the **Properties** dialog for the SR. In the **Read Caching** tab, you can select to enable or disable read caching.

For more information, see Changing SR Properties.

Limitations

- Read caching is available only for NFS, EXT3/EXT4, SMB, and GFS2 SRs. It is not available for other SR types.
- Read caching only applies to read-only VDIs and VDI parents. These VDIs exist where VMs are created from 'Fast Clone'or disk snapshots. The greatest performance improvements can be seen when many VMs are cloned from a single 'golden'image.
- Performance improvements depend on the amount of free memory available in the host's Control Domain (dom0). Increasing the amount of dom0 memory allows more memory to be allocated to the read-cache. For information on how to configure dom0 memory, see CTX134951.
- When memory read caching is turned on, a cache miss causes I/O to become serialized. This can sometimes be more expensive than having read caching turned off, because with read caching turned off I/O can be parallelized. To reduce the impact of cache misses, increase the amount of available dom0 memory or disable read caching for the SR.

Comparison with IntelliCache

IntelliCache and memory based read caching are to some regards complementary. IntelliCache not only caches on a different tier, but it also caches writes in addition to reads. IntelliCache caches reads from the network onto a local disk. In-memory read caching caches the reads from network or disk into host memory. The advantage of in-memory read caching, is that memory is still an order of magnitude faster than a solid-state disk (SSD). Performance in boot storms and other heavy I/O situations improves.

Both read-caching and IntelliCache can be enabled simultaneously. In this case, IntelliCache caches the reads from the network to a local disk. Reads from that local disk are cached in memory with read caching.

Set the read cache size

The read cache performance can be optimized, by giving more memory to XenServer's control domain (dom0).

> **Important:**
>
> Set the read cache size on ALL hosts in the pool individually for optimization. Any subsequent changes to the size of the read cache must also be set on all hosts in the pool.

On the XenServer host, open a local shell and log on as root.

To set the size of the read cache, run the following command:

```
/opt/xensource/libexec/xen-cmdline --set-xen dom0_mem=nnM,max:nnM
<!--NeedCopy-->
```

Set both the initial and maximum values to the same value. For example, to set dom0 memory to 20,480 MiB:

```
/opt/xensource/libexec/xen-cmdline --set-xen dom0_mem=20480M,max:20480M
<!--NeedCopy-->
```

> **Important:**
>
> Reboot all hosts after changing the read cache size.

How to view the current dom0 memory allocation?

To view the current dom0 memory settings, enter:

```
free -m
<!--NeedCopy-->
```

The output of `free -m` shows the current dom0 memory settings. The value may be less than expected due to various overheads. The example table below shows the output from a host with dom0 set to 2.6 GiB

```
|                      | Total  | Used | Free  | Shared | Buffer/cache | Available |
|----------------------|--------|------|-------|--------|--------------|----------
| Mem:                 | 2450   | 339  | 1556  | 9      | 554          | 2019      |
| Swap:                | 1023   | 0    | 1023  |        |              |           |
<!--NeedCopy-->
```

What Range of Values Can be Used? As the XenServer Control Domain (dom0) is 64-bit, large values can be used, for example 32,768 MiB. However, we recommend that you **do not reduce the dom0 memory below 1 GiB**.

XenCenter display notes

The entire host's memory can be considered to comprise the Xen hypervisor, dom0, VMs, and free memory. Even though dom0 and VM memory is usually of a fixed size, the Xen hypervisor uses a variable amount of memory. The amount of memory used depends on various factors. These factors include the number of VMs running on the host at any time and how those VMs are configured. It is not possible to limit the amount of memory that Xen uses. Limiting the amount of memory can cause Xen to run out of memory and prevent new VMs from starting, even when the host had free memory.

To view the memory allocated to a host, in XenCenter select the host, and then click the **Memory** tab.

The XenServer field displays the *sum* of the memory allocated to dom0 *and* Xen memory. Therefore, the amount of memory displayed might be higher than specified by the administrator. The memory size can vary when starting and stopping VMs, even when the administrator has set a fixed size for dom0.

Graphics overview

December 21, 2023

This section provides an overview of the virtual delivery of 3D professional graphics applications and workstations in XenServer. The offerings include GPU Pass-through (for NVIDIA, AMD, and Intel GPUs) and hardware-based GPU sharing with NVIDIA vGPU™ and Intel GVT-g™.

Graphics Virtualization is available for XenServer Premium Edition customers. To learn more about XenServer editions, and to find out how to upgrade, visit the XenServer website. For more information, see Licensing.

GPU pass-through

In a virtualized system, most of the physical system components are shared. These components are represented as multiple virtual instances to multiple clients by the hypervisor. A pass-through GPU is not abstracted at all, but remains one physical device. Each hosted virtual machine (VM) gets its own dedicated GPU, eliminating the software abstraction and the performance penalty that goes with it.

XenServer allows you to assign a physical GPU (in the XenServer host) to a Windows or Linux VM running on the same host. This GPU pass-through feature is intended for graphics power users, such as CAD designers.

Shared GPU (vGPU)

Shared GPU (vGPU) allows one physical GPU to be used by multiple VMs concurrently. Because a portion of a physical GPU is used, performance is greater than emulated graphics, and there is no need for one card per VM. This feature enables resource optimization, boosting the performance of the VM. The graphics commands of each virtual machine are passed directly to the GPU, without translation by the hypervisor.

Multiple shared GPU (vGPU)

Multiple vGPU enables multiple virtual GPUs to be used concurrently by a single VM. Only certain vGPU profiles can be used and all vGPUs attached to a single VM must be of the same type. These additional vGPUs can be used to perform computational processing. For more information about the number of vGPUs supported for a single VM, see Configuration Limits.

This feature is only available for NVIDIA GPUs. For more information about the physical GPUs that support the multiple vGPU feature, see the NVIDIA documentation.

Vendor support

The following table lists guest support for the GPU pass-through, shared GPU (vGPU), and multiple shared GPU (vGPU) features:

	GPU pass-through for Windows VMs	GPU pass-through for Linux VMs	Shared GPU (vGPU) for Windows VMs	Shared GPU (vGPU) for Linux VMs	Multiple shared GPU (vGPU) for Windows VMs	Multiple shared GPU (vGPU) for Linux VMs
AMD	YES					
Intel	YES		YES (deprecated)			
NVIDIA	YES	YES	YES	YES	YES (see note)	YES (see note)

> **Note:**
>
> - Only some of the guest operating systems support multiple vGPU. For more information, see Guest support and constraints.
> - Only some of the guest operating systems support vGPU live migration. For more information, see Vendor support.

You might need a vendor subscription or a license depending on the graphics card used.

vGPU live migration

vGPU live migration enables a VM that uses a virtual GPU to perform live migration, storage live migration, or VM suspend. VMs with vGPU live migration capabilities can be migrated to avoid downtime.

vGPU live migration also enables you to perform rolling pool upgrades on pools that host vGPU-enabled VMs. For more information, see Rolling pool upgrades.

To use vGPU live migration or VM suspend, your VM must run on a graphics card that supports this feature. Your VM must also have the supported drivers from the GPU vendor installed.

> **Warning:**
>
> The size of the GPU state in the NVIDIA driver can cause a downtime of 5 seconds or more during vGPU live migration.

The following restrictions apply when using vGPU live migration:

- Live migration is not compatible with GPU Pass-through.
- VMs must have the appropriate vGPU drivers installed to be supported with any vGPU live migration features. The in-guest drivers must be installed for all guests using the vGPU feature.

- Reboot and shutdown operations on a VM are not supported while a migration is in progress. These operations can cause the migration to fail.
- Linux VMs are not supported with any vGPU live migration features.
- Live migration by the Workload Balancing appliance is not supported for vGPU-enabled VMs. The Workload Balancing appliance cannot do capacity planning for VMs that have a vGPU attached.
- After migrating a VM using vGPU live migration, the guest VNC console might become corrupted. Use ICA, RDP, or another network-based method for accessing VMs after a vGPU live migration has been performed.
- VDI migration uses live migration, therefore requires enough vGPU space on the host to make a copy of the vGPU instance on the host. If the physical GPUs are fully used, VDI migration might not be possible.

Vendor support

The following table lists support for vGPU live migration:

	GPU pass-through for Windows VMs	GPU pass-through for Linux VMs	Shared GPU (vGPU) for Windows VMs	Shared GPU (vGPU) for Linux VMs	Multiple shared GPU (vGPU) for Windows VMs	Multiple shared GPU (vGPU) for Linux VMs
NVIDIA			YES		YES	

For more information about the graphics cards that support this feature, see the vendor-specific sections of this guide. Customers might need a vendor subscription or a license depending on the graphics card used.

Guest support and constraints

XenServer supports the following guest operating systems for virtual GPUs.

NVIDIA vGPU

Operatings systems marked with an asterisk (*) also support multiple vGPU.

Windows guests:

- Windows 10 (64-bit) *
- Windows 11 (64-bit) *
- Windows Server 2016 (64-bit) *
- Windows Server 2019 (64-bit) *
- Windows Server 2022 (64-bit) *

Linux guests:

- RHEL 7 *
- RHEL 8 *
- RHEL 9 *
- CentOS 7
- CentOS Stream 9
- Ubuntu 18.04 * (deprecated)
- Ubuntu 20.04 *
- Ubuntu 22.04 *
- Rocky Linux 8 *
- Rocky Linux 9 *

Intel GVT-g (deprecated)

Windows guests:

- Windows 10 (64-bit)
- Windows Server 2016 (64-bit)

Constraints

- VMs with a virtual GPU are not supported with Dynamic Memory Control.
- XenServer automatically detects and groups identical physical GPUs across hosts in the same pool. If assigned to a group of GPUs, a VM can be started on any host in the pool that has an available GPU in the group.
- All graphics solutions (NVIDIA vGPU, Intel GVT-d, Intel GVT-G, and vGPU pass-through) can be used in an environment that uses high availability. However, VMs that use these graphics solutions cannot be protected with high availability. These VMs can be restarted on a best-effort basis while there are hosts with the appropriate free resources.

Prepare host for graphics

April 10, 2024

This section provides step-by-step instructions on how to prepare XenServer for supported graphical virtualization technologies. The offerings include NVIDIA vGPU, Intel GVT-d, and Intel GVT-g.

NVIDIA vGPU

NVIDIA vGPU enables multiple Virtual Machines (VM) to have simultaneous, direct access to a single physical GPU. It uses NVIDIA graphics drivers deployed on non-virtualized Operating Systems. NVIDIA physical GPUs can support multiple virtual GPU devices (vGPUs). To provide this support, the physical GPU must be under the control of NVIDIA Virtual GPU Manager running in XenServer Control Domain (dom0). The vGPUs can be assigned directly to VMs.

VMs use virtual GPUs like a physical GPU that the hypervisor has passed through. An NVIDIA driver loaded in the VM provides direct access to the GPU for performance critical fast paths. It also provides a paravirtualized interface to the NVIDIA Virtual GPU Manager.

> **Important:**
>
> To ensure that you always have the latest security and functional fixes, ensure that you install the latest NVIDIA vGPU software package for XenServer (consisting of the NVIDIA Virtual GPU Manager for XenServer and NVIDIA drivers) and keep it updated to the latest version provided by NVIDIA. For more information, see the NVIDIA documentation.
>
> The latest NVIDIA drivers are available from the NVIDIA Application Hub.

NVIDIA vGPU is compatible with the HDX 3D Pro feature of Citrix Virtual Apps and Desktops or Citrix DaaS. For more information, see HDX 3D Pro.

Licensing note

NVIDIA vGPU is available for XenServer Premium Edition customers. To learn more about XenServer editions, and to find out how to upgrade, visit the XenServer website. For more information, see Licensing.

Depending on the NVIDIA graphics card used, you might need NVIDIA subscription or a license.

For information on licensing NVIDIA cards, see the NVIDIA website.

Available NVIDIA vGPU types

NVIDIA GRID cards contain multiple Graphics Processing Units (GPU). For example, TESLA M10 cards contain four GM107GL GPUs, and TESLA M60 cards contain two GM204GL GPUs. Each physical GPU can host several different types of virtual GPU (vGPU). vGPU types have a fixed amount of frame buffer, number of supported display heads and maximum resolutions, and are targeted at different classes of workload.

For a list of the most recently supported NVIDIA cards, see the Hardware Compatibility List and the NVIDIA product information.

> **Note:**
>
> The vGPUs hosted on a physical GPU at the same time **must all be of the same type.** However, there is no corresponding restriction for physical GPUs on the same card. This restriction is automatic and can cause unexpected capacity planning issues.

NVIDIA vGPU system requirements

- NVIDIA GRID card:
  - For a list of the most recently supported NVIDIA cards, see the Hardware Compatibility List and the NVIDIA product information.
- Depending on the NVIDIA graphics card used, you might need an NVIDIA subscription or a license. For more information, see the NVIDIA product information.
- Depending on the NVIDIA graphics card, you might need to ensure that the card is set to the correct mode. For more information, see the NVIDIA documentation.
- XenServer Premium Edition.
- A host capable of hosting XenServer and the supported NVIDIA cards.
- NVIDIA vGPU software package for XenServer, consisting of the NVIDIA Virtual GPU Manager for XenServer, and NVIDIA drivers.

  > **Note:**
  >
  > Review the NVIDIA Virtual GPU software documentation available from the NVIDIA website. Register with NVIDIA to access these components.

- To run Citrix Virtual Desktops with VMs running NVIDIA vGPU, you also need: Citrix Virtual Desktops 7.6 or later, full installation.
- For NVIDIA Ampere vGPUs and all future generations, you must enable SR-IOV in your system firmware.

vGPU live migration

XenServer enables the use of live migration, storage live migration, and the ability to suspend and resume for NVIDIA vGPU-enabled VMs.

To use the vGPU live migration, storage live migration, or Suspend features, satisfy the following requirements:

- An NVIDIA GRID card, Maxwell family or later.
- An NVIDIA Virtual GPU Manager for XenServer with live migration enabled. For more information, see the NVIDIA Documentation.
- A Windows VM that has NVIDIA live migration-enabled vGPU drivers installed.

vGPU live migration enables the use of live migration within a pool, live migration between pools, storage live migration, and Suspend/Resume of vGPU-enabled VMs.

Preparation overview

1. Install XenServer
2. Install the NVIDIA Virtual GPU Manager for XenServer
3. Restart the XenServer host

Installation on XenServer

XenServer is available for download from the XenServer Downloads page.

Install the following:

- **XenServer Base Installation ISO**
- **XenCenter Windows Management Console**

For more information, see Install.

Licensing note

vGPU is available for XenServer Premium Edition customers. To learn more about XenServer editions, and to find out how to upgrade, visit the XenServer website. For more information, see Licensing.

Depending on the NVIDIA graphics card used, you might need NVIDIA subscription or a license. For more information, see NVIDIA product information.

For information about licensing NVIDIA cards, see the NVIDIA website.

Install the NVIDIA vGPU Manager for XenServer

Install the NVIDIA Virtual GPU software that is available from NVIDIA. The NVIDIA Virtual GPU software consists of:

- NVIDIA Virtual GPU Manager
- Windows Display Driver (The Windows display driver depends on the Windows version)

The **NVIDIA Virtual GPU Manager** runs in the XenServer Control Domain (dom0). It is provided as either a supplemental pack or an RPM file. For more information about installation, see the NVIDIA virtual GPU Software documentation.

The Update can be installed in one of the following methods:

- Use XenCenter (**Tools** > **Install Update** > **Select update or supplemental pack from disk**)
- Use the xe CLI command `xe-install-supplemental-pack`.

> **Note:**
>
> If you are installing the NVIDIA Virtual GPU Manager using an RPM file, ensure that you copy the RPM file to dom0 and then install.

1. Use the rpm command to install the package:

   ```
   rpm -iv <vgpu_manager_rpm_filename>
   <!--NeedCopy-->
   ```

2. Restart the XenServer host:

   ```
   shutdown -r now
   <!--NeedCopy-->
   ```

3. After you restart the XenServer host, verify that the software has been installed and loaded correctly by checking the NVIDIA kernel driver:

   ```
   [root@xenserver ~]#lsmod |grep nvidia
       nvidia               8152994 0
   <!--NeedCopy-->
   ```

4. Verify that the NVIDIA kernel driver can successfully communicate with the NVIDIA physical GPUs in your host. Run the `nvidia-smi` command to produce a listing of the GPUs in your platform similar to:

   ```
   [root@xenserver ~]# nvidia-smi

       Thu Jan 26 13:48:50 2017
       +-----------------------------------------------------------+|
       NVIDIA-SMI 367.64   Driver Version: 367.64                 |
       -------------------------------+-----------------------+
   ```

```
      GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr.
           ECC|
     Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util
         Compute M.|
     ===============================+======================+==================

     |   0  Tesla M60          On   | 0000:05:00.0     Off|     Off |
     | N/A   33C    P8    24W / 150W |   7249MiB /  8191MiB |       0%
              Default    |
     +-------------------------------+----------------------+------------------

     |   1  Tesla M60          On   | 0000:09:00.0     Off |    Off |
     | N/A   36C    P8    24W / 150W |   7249MiB /  8191MiB |       0%
              Default    |
     +-------------------------------+----------------------+------------------

     |   2  Tesla M60          On   | 0000:85:00.0     Off |    Off |
     | N/A   36C    P8    23W / 150W |     19MiB /  8191MiB |       0%
              Default    |
     +-------------------------------+----------------------+------------------

     |   3  Tesla M60          On   | 0000:89:00.0     Off |    Off |
     | N/A   37C      P8    23W / 150W |     14MiB /  8191MiB |     0%
              Default    |
     +-------------------------------+----------------------+------------------

     +-----------------------------------------------------------------------------

     | Processes:                           GPU Memory |
     | GPU       PID   Type   Process name      Usage     |
     |=============================================================================

     | No running compute processes found |
     +-----------------------------------------------------------------------------

<!--NeedCopy-->
```

Note:

When using NVIDIA vGPU with XenServer servers that have more than 768 GB of RAM, add the parameter `iommu=dom0-passthrough` to the Xen command line:

a) Run the following command in the control domain (Dom0):

   `/opt/xensource/libexec/xen-cmdline --set-xen iommu=dom0-passthrough`

b) Restart the host.

Intel GVT-d and GVT-g

XenServer supports Intel's virtual GPU (GVT-g), a graphics acceleration solution that requires no additional hardware. It uses the Intel Iris Pro feature embedded in certain Intel processors, and a standard Intel GPU driver installed within the VM.

To ensure that you always have the latest security and functional fixes, ensure that you install any updates provided by Intel for the drivers on your VMs and the firmware on your host.

Intel GVT-d and GVT-g are compatible with the HDX 3D Pro features of Citrix Virtual Apps and Desktops or Citrix DaaS. For more information, see HDX 3D Pro.

> **Note:**
>
> Because the Intel Iris Pro graphics feature is embedded within the processors, CPU-intensive applications can cause power to be diverted from the GPU. As a result, you might not experience full graphics acceleration as you do for purely GPU-intensive workloads.

Intel GVT-g system requirements and configuration (deprecated)

To use Intel GVT-g, your XenServer host must have the following hardware:

- A CPU that has Iris Pro graphics. This CPU must be listed as supported for Graphics on the Hardware Compatibility List
- A motherboard that has a graphics-enabled chipset. For example, C226 for Xeon E3 v4 CPUs or C236 for Xeon E3 v5 CPUs.

> **Note:**
>
> Ensure that you restart the hosts when switching between Intel GPU pass-through (GVT-d) and Intel Virtual GPU (GVT-g).

When configuring Intel GVT-g, the number of Intel virtual GPUs supported on a specific XenServer host depends on its GPU bar size. The GPU bar size is called the 'Aperture size'in the system firmware. We recommend that you set the Aperture size to 1,024 MB to support a maximum of seven virtual GPUs per host.

If you configure the Aperture size to 256 MB, only one VM can start on the host. Setting it to 512 MB can result in only three VMs being started on the XenServer host. An Aperture size higher than 1,024 MB is not supported and **does not** increase the number of VMs that start on a host.

Enable Intel GPU pass-through

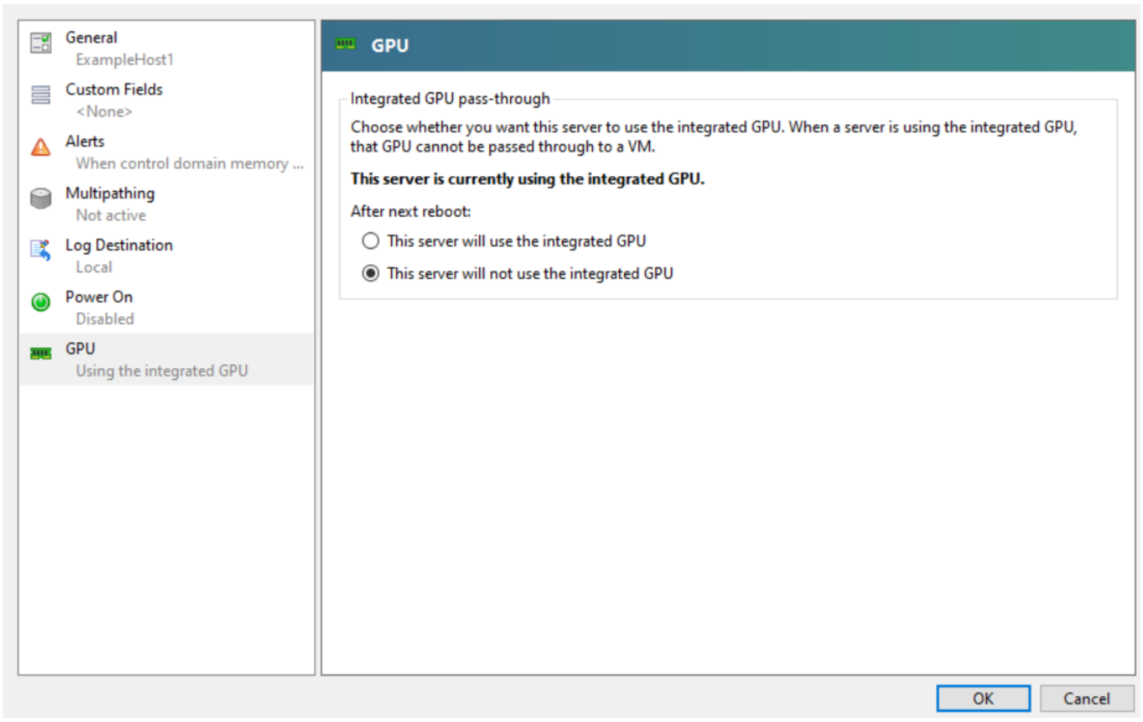
XenServer supports the GPU pass-through feature for Windows VMs using an Intel integrated GPU device.

- For more information on Windows versions supported with Intel GPU pass-through, see Graphics.
- For more information on supported hardware, see the Hardware Compatibility List.

When using Intel GPU on Intel servers, the XenServer server's Control Domain (dom0) has access to the integrated GPU device. In such cases, the GPU is available for pass-through. To use the Intel GPU Pass-through feature on Intel servers, disable the connection between dom0 and the GPU before passing through the GPU to the VM.

To disable this connection, complete the following steps:

1. On the **Resources** pane, choose the XenServer host.
2. On the **General** tab, click **Properties**, and in the left pane, click **GPU**.
3. In the **Integrated GPU passthrough** section, select **This server will not use the integrated GPU**.

   Integrated GPU pass-through

   Choose whether you want this server to use the integrated GPU. When a server is using the integrated GPU, that GPU cannot be passed through to a VM.

   This server is currently using the integrated GPU.

   After next reboot:

   ○ This server will use the integrated GPU

   ◉ This server will not use the integrated GPU

   This step disables the connection between dom0 and the Intel integrated GPU device.

4. Click **OK**.
5. Restart the XenServer host for the changes to take effect.

   The Intel GPU is now visible on the GPU type list during new VM creation, and on the VM's **Properties** tab.

> **Note:**
>
> The XenServer host's external console output (for example, VGA, HDMI, DP) will not be available after disabling the connection between dom0 and the GPU.

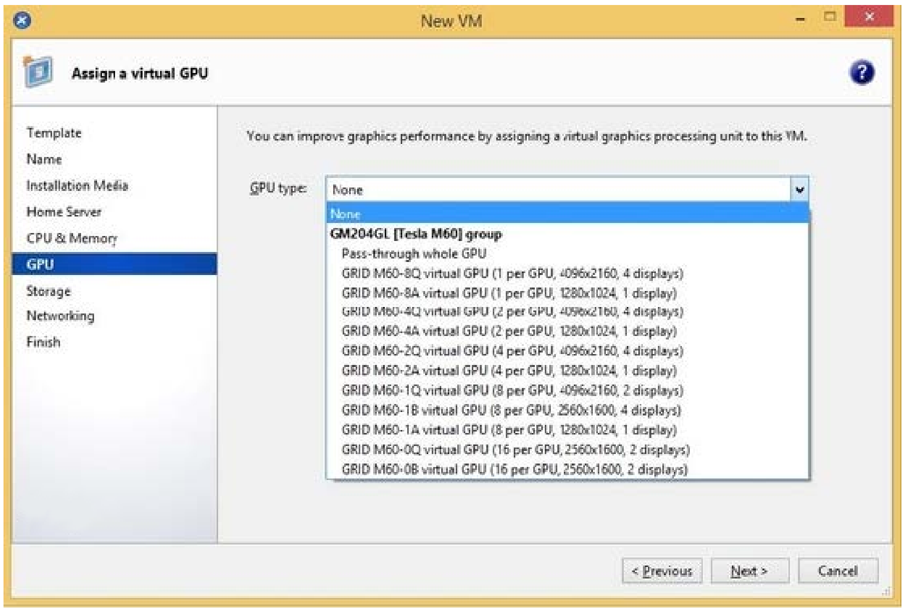
Create vGPU enabled VMs

February 9, 2024

This section provides step-by-step instructions on how to create a virtual GPU or GPU pass-through enabled VM.

> **Note:**
>
> If you are using the Intel GPU Pass-through feature, first see the section *Enabling Intel GPU Pass-through* for more configuration, and then complete the following steps.

1. Create a VM using XenCenter. Select the host on the Resources pane and then select **New VM** on the VM menu.
2. Follow the instructions on the **New VM** configuration and select the **Installation Media**, **Home Server**, and **CPU & Memory**.
3. GPU-enabled hosts display a **GPU** configuration page:



You can improve graphics performance by assigning a virtual graphics processing unit to this VM.

GPU type: None

None
GM204GL [Tesla M60] group
Pass-through whole GPU
GRID M60-8Q virtual GPU (1 per GPU, 4096x2160, 4 displays)
GRID M60-8A virtual GPU (1 per GPU, 1280x1024, 1 display)
GRID M60-4Q virtual GPU (2 per GPU, 4096x2160, 4 displays)
GRID M60-4A virtual GPU (2 per GPU, 1280x1024, 1 display)
GRID M60-2Q virtual GPU (4 per GPU, 4096x2160, 4 displays)
GRID M60-2A virtual GPU (4 per GPU, 1280x1024, 1 display)
GRID M60-1Q virtual GPU (8 per GPU, 4096x2160, 2 displays)
GRID M60-1B virtual GPU (8 per GPU, 2560x1600, 4 displays)
GRID M60-1A virtual GPU (8 per GPU, 1280x1024, 1 display)
GRID M60-0Q virtual GPU (16 per GPU, 2560x1600, 2 displays)
GRID M60-0B virtual GPU (16 per GPU, 2560x1600, 2 displays)

4. Click **Add**. From the **GPU Type** list, select either **Pass-through whole GPU**, or a virtual GPU type.

Unavailable virtual GPU types are grayed-out.

If you want to assign multiple vGPUs to your VM, ensure that you select a vGPU type that supports multiple vGPU. Repeat this step to add more vGPUs of the same type.

5. Click **Next** to configure **Storage** and then **Networking**.
6. After you complete your configuration, click **Create Now**.

Install the XenServer VM Tools

Without the optimized networking and storage drivers provided by the XenServer VM Tools, remote graphics applications running on NVIDIA vGPU do **not** deliver maximum performance.

- If your VM is a Windows VM, you must install the XenServer VM Tools for Windows on your VM. For more information, see Install XenServer VM Tools for Windows.
- If your VM is a Linux VM, you can install the XenServer VM Tools for Linux on your VM. For more information, see Install XenServer VM Tools for Linux.

Install the in-guest drivers

When viewing the VM console in XenCenter, the VM typically boots to the desktop in VGA mode with 800 x 600 resolution. The standard Windows screen resolution controls can be used to increase the resolution to other standard resolutions. (**Control Panel > Display > Screen Resolution**)

> **Note:**
>
> When using GPU pass-through, we recommend that you install the in-guest drivers through RDP or VNC over the network. That is, not through XenCenter.

To ensure that you always have the latest security and functional fixes, ensure that you always take the latest updates to your in-guest drivers.

Install the NVIDIA drivers

To enable vGPU operation (as for a physical NVIDIA GPU), install NVIDIA drivers into the VM.

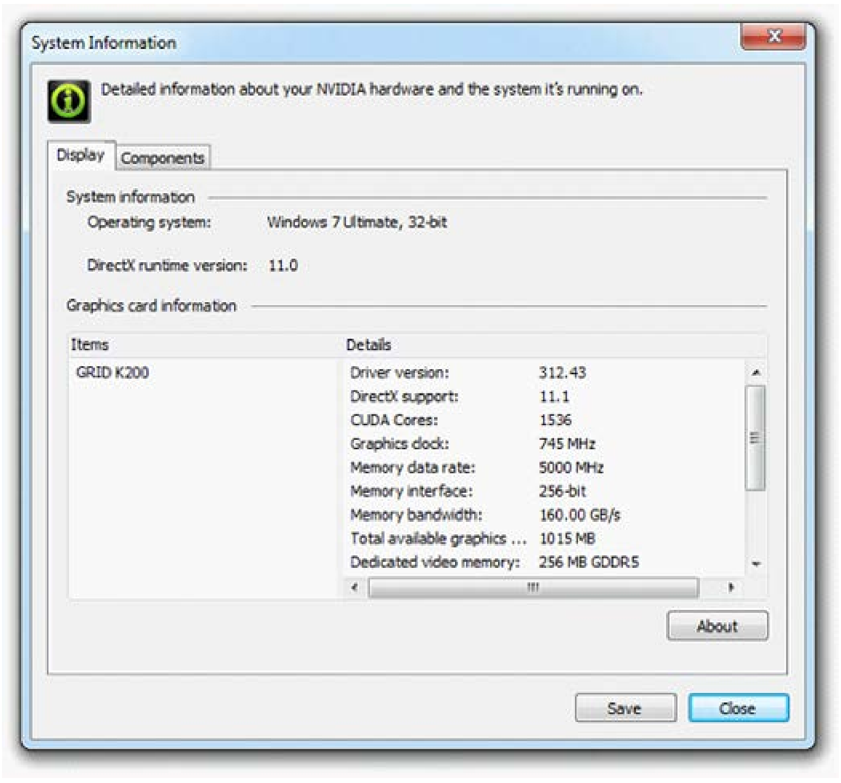
The following section provides an overview of the procedure. For detailed instructions, see the NVIDIA virtual GPU Software documentation.

1. Start the VM. In the **Resources** pane, right-click on the VM, and click **Start**.

   During this start process, XenServer dynamically allocates a vGPU to the VM.

2. Follow the Windows operating system installation screens.

3. After the operating system installation completes, restart the VM.
4. Install the appropriate driver for the GPU inside the guest. The following example shows the specific case for in guest installation of the NVIDIA GRID drivers.
5. Copy the 64-bit NVIDIA Windows driver package to the VM, open the zip file, and run setup.exe.
6. Follow the installer steps to install the driver.
7. After the driver installation has completed, you might be prompted to reboot the VM. Select **Restart Now** to restart the VM immediately, alternatively, exit the installer package, and restart the VM when ready. When the VM starts, it boots to a Windows desktop.
8. To verify that the NVIDIA driver is running, right-click on the desktop and select **NVIDIA Control Panel**.
9. In the NVIDIA Control Panel, select **System Information**. This interface shows the GPU Type in use by the VM, its features, and the NVIDIA driver version in use:



Detailed information about your NVIDIA hardware and the system it's running on.

System information

Operating system:	Windows 7 Ultimate, 32-bit
DirectX runtime version:	11.0

Graphics card information

Items	Details	
GRID K200	Driver version:	312.43
	DirectX support:	11.1
	CUDA Cores:	1536
	Graphics clock:	745 MHz
	Memory data rate:	5000 MHz
	Memory interface:	256-bit
	Memory bandwidth:	160.00 GB/s
	Total available graphics ...	1015 MB
	Dedicated video memory:	256 MB GDDR5

> **Note:**
>
> Depending on the NVIDIA graphics card used, you might need an NVIDIA subscription or a license. For more information, see the NVIDIA product information.

The VM is now ready to run the full range of DirectX and OpenGL graphics applications supported by the GPU.

For Linux VMs

In your Linux VM, install the driver as instructed by the NVIDIA User Guides.

If your Linux VM boots in UEFI Secure Boot mode, you might be required to take additional steps to sign the driver. For more information, see Install third-party drivers on your Secure Boot Linux VM.

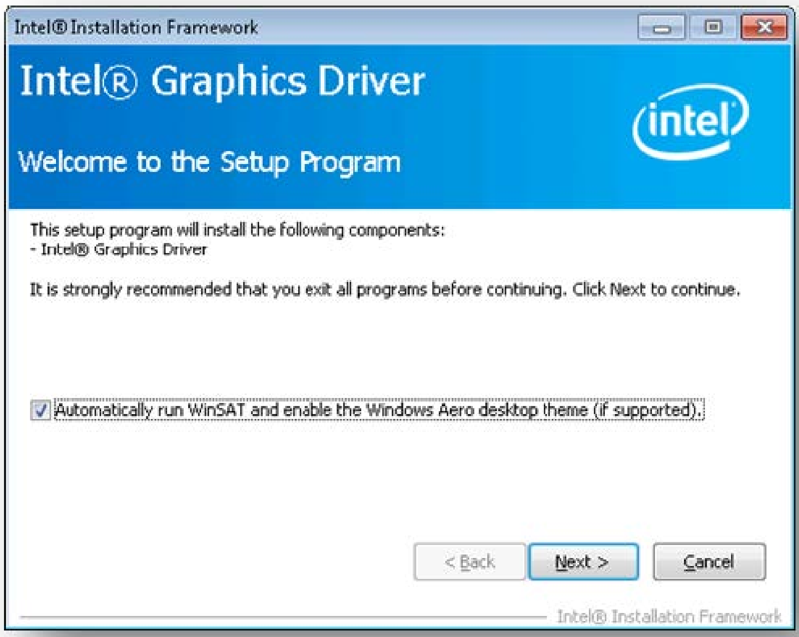
Install the Intel drivers (deprecated)

To enable GPU operation, install Intel drivers into the VM.

1. Start the **VM**. In the **Resources** pane, right-click on the VM, and click **Start**.

   During this boot process, XenServer dynamically allocates a GPU to the VM.

2. Follow the Windows operating system installation screens.
3. After the operating system installation completes, reboot the VM.
4. Copy the 64-bit Intel Windows driver (Intel Graphics Driver) to the VM.
5. Run the **Intel Graphics Driver** setup program
6. Select **Automatically run WinSAT**, and then click **Next**.


Intel® Installation Framework
Intel® Graphics Driver
Welcome to the Setup Program
intel
< Back
Next >
Cancel
Intel® Installation Framework

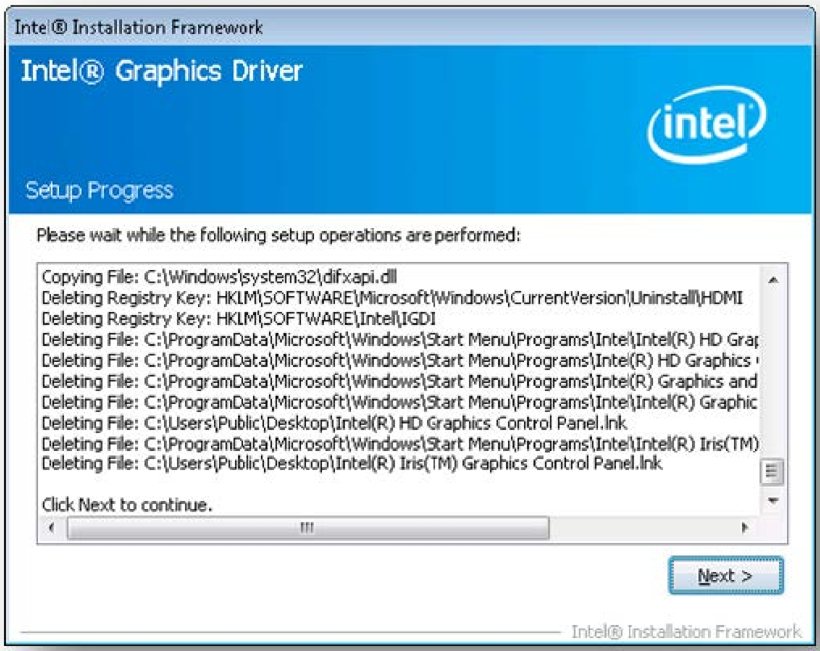
This setup program will install the following components:
- Intel® Graphics Driver

It is strongly recommended that you exit all programs before continuing. Click Next to continue.

[✓] Automatically run WinSAT and enable the Windows Aero desktop theme (if supported).

7. To accept the License Agreement, click **Yes**, and on the Readme File Information screen, click **Next**.

8. Wait until the setup operations complete. When you are prompted, click **Next**.



Please wait while the following setup operations are performed:

```
Copying File: C:\Windows\system32\difxapi.dll
Deleting Registry Key: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\HDMI
Deleting Registry Key: HKLM\SOFTWARE\Intel\IGDI
Deleting File: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Intel\Intel(R) HD Grap
Deleting File: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Intel(R) HD Graphics
Deleting File: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Intel(R) Graphics and
Deleting File: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Intel\Intel(R) Graphic
Deleting File: C:\Users\Public\Desktop\Intel(R) HD Graphics Control Panel.lnk
Deleting File: C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Intel\Intel(R) Iris(TM)
Deleting File: C:\Users\Public\Desktop\Intel(R) Iris(TM) Graphics Control Panel.lnk

Click Next to continue.
```

9. To complete the installation, you are prompted to restart the VM. Select **Yes**, I want to restart this computer now, and click **Finish**.

10. After the VM restarts, check that graphics are working correctly. Open the Windows Device Manager, expand **Display adapters**, and ensure that the Intel Graphics Adapter does not have any warning symbols.

    > **Note:**
    >
    > You can obtain the latest drivers from the Intel website.

Memory usage

March 18, 2024

Two components contribute to the memory footprint of the XenServer host. First, the memory consumed by the Xen hypervisor itself. Second, there is the memory consumed by the *Control Domain* of the host. Also known as 'Domain0', or 'dom0', the control domain is a secure, privileged Linux VM that runs the XenServer management toolstack (XAPI). Besides providing XenServer management functions, the control domain also runs the driver stack that provides user created VM access to physical devices.

Control domain memory

The amount of memory allocated to the control domain is adjusted automatically and is based on the amount of physical memory on the physical host. By default, XenServer allocates **1 GiB plus 5% of the total physical memory** to the control domain, up to an initial maximum of 8 GiB.

> **Note:**
>
> The amount reported in the XenServer section in XenCenter includes the memory used by the control domain (dom0), the Xen hypervisor itself, and the crash kernel. Therefore, the amount of memory reported in XenCenter can exceed these values. The amount of memory used by the hypervisor is larger for hosts using more memory.

Change the amount of memory allocated to the control domain

You can change the amount of memory allocated to dom0 by using XenCenter or by using the command line. If you increase the amount of memory allocated to the control domain beyond the amount allocated by default, this action results in less memory being available to VMs.

You might need to increase the amount of memory assigned to the control domain of a XenServer host in the following cases:

- You are running many VMs on the host
- You are using PVS-Accelerator
- You are using read caching

> **Important:**
>
> If you are using a GFS2 SR and any of these cases also applies to your environment, you must increase the amount of control domain memory. Insufficient control domain memory can cause network instability, which can cause problems for clustered pools with GFS2 SRs.

The amount of memory to allocate to the control domain depends on your environment and the requirements of your VMs.

You can monitor the following metrics to judge whether the amount of control domain memory is appropriate for your environment and what effects any changes you make have:

- **Swap activity:** If the control domain is swapping, increase the control domain memory.

- **Tapdisk mode:** You can monitor whether your tapdisks are in low-memory mode from within the XenCenter **Performance** tab for the host. Select **Actions** > **New Graph** and choose the **Tapdisks in low memory mode** graph. If a tapdisk is in low-memory mode, increase the control domain memory.
- **Pagecache pressure:** Use the `top` command to monitor the `buff/cache` metric. If this number becomes too low, you might want to increase the control domain memory.

Changing the dom0 memory by using XenCenter

For information about changing the dom0 memory by using XenCenter, see Changing the Control Domain Memory in the XenCenter documentation.

> **Note:**
>
> You cannot use XenCenter to reduce dom0 memory below the value that was initially set during XenServer installation. To make this change you must use the command line.

Changing the dom0 memory by using the command line

> **Note:**
>
> On hosts with smaller memory (less than 16 GiB), you might want to reduce the memory allocated to the Control Domain to lower than the installation default value. You can use the command line to make this change. However, we recommend that you **do not reduce the dom0 memory below 1 GiB** and that you do this operation under the guidance of the Support Team.

1. On the XenServer host, open a local shell and log on as root.
2. Type the following:

   ```
   /opt/xensource/libexec/xen-cmdline --set-xen dom0_mem=<nn>M,max:<
       nn>M
   <!--NeedCopy-->
   ```

   Where `<nn>` represents the amount of memory, in MiB, to be allocated to dom0.
3. Restart the XenServer host using XenCenter or the `reboot` command on the XenServer console.

   When the host restarts, on the XenServer console, run the `free` command to verify the new memory settings.

How much memory is available to VMs?

To find out how much host memory is available to be assigned to VMs, find the value of the free memory of the host by running `memory-free`. Then type the command `vm-compute-maximum`

`-memory` to get the actual amount of free memory that can be allocated to the VM. For example:

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

Monitor and manage your deployment

March 22, 2024

XenServer provides detailed monitoring of performance metrics. These metrics include CPU, memory, disk, network, C-state/P-state information, and storage. Where appropriate, these metrics are available on a per host and a per VM basis. These metrics are available directly, or can be accessed and viewed graphically in XenCenter or other third-party applications.

XenServer also provides system and performance alerts. Alerts are notifications that occur in response to selected system events. These notifications also occur when one of the following values goes over a specified threshold on a managed host, VM, or storage repository: CPU usage, network usage, memory usage, control domain memory usage, storage throughput, or VM disk usage. You can configure the alerts by using the xe CLI or by using XenCenter. To create notifications based on any of the available Host or VM performance metrics see Performance alerts.

Monitor XenServer performance

Customers can monitor the performance of their XenServer hosts and Virtual Machines (VMs) using the metrics exposed through Round Robin Databases (RRDs). These metrics can be queried over HTTP or through the RRD2CSV tool. In addition, XenCenter uses this data to produce system performance graphs. For more information, see Analyze and visualize metrics.

The following tables list all of the available host and VM metrics.

> **Notes:**
>
> - Latency over a period is defined as the average latency of operations during that period.
> - The availability and utility of certain metrics are SR and CPU dependent.
> - Performance metrics are not available for GFS2 SRs and disks on those SRs.

Available host metrics

Metric Name	Description	Condition	XenCenter Name
avgqu_sz_<sr-uuid-**short**>	Average I/O queue size (requests).	At least one plugged VBD in SR <sr-uuid-**short**> on the host	sr-uuid-**short** Queue Size
cpu<cpu>-C<cstate>	Time CPU cpu spent in C-state cstate in milliseconds.	C-state exists on CPU	CPU cpu C-state cstate
cpu<cpu>-P<pstate>	Time CPU cpu spent in P-state pstate in milliseconds.	P-state exists on CPU	CPU cpu P-state pstate
cpu<cpu>	Utilization of physical CPU cpu (fraction). Enabled by default.	CPU cpu exists	CPU cpu
cpu_avg	Mean utilization of physical CPUs (fraction). Enabled by default.	None	Average CPU
hostload	Host load per physical CPU, where load refers to the number of vCPU(s) in a running or runnable state.	None	Host CPU Load
inflight_<sr-uuid-**short**>	Number of I/O requests currently in flight. Enabled by default.	At least one plugged VBD in SR sr on the host	sr Inflight Requests
io_throughput_read_<sr-uuidshort>	Data read from SR (MiB/s).	At least one plugged VBD in SR sr on the host	sr Read Throughput
io_throughput_write_<sr-uuidshort>	Data written to the SR (MiB/s).	At least one plugged VBD in SR sr on the host	sr Write Throughput
io_throughput_total_<sr-uuidshort>	All SR I/O (MiB/s).	At least one plugged VBD in SR sr on the host	sr Total Throughput
iops_read_<sr-uuid-**short**>	Read requests per second.	At least one plugged VBD in SR sr on the host	sr Read IOPS

Metric Name	Description	Condition	XenCenter Name
`iops_write_<sr-uuid-short>`	Write requests per second.	At least one plugged VBD in SR `sr` on the host	`sr` Write IOPS
`iops_total_<sr-uuid-short>`	I/O requests per second.	At least one plugged VBD in SR `sr` on the host	`sr` Total IOPS
`iowait_<sr-uuid-short>`	Percentage of the time waiting for I/O.	At least one plugged VBD in SR `sr` on the host	`sr` IO Wait
`latency_<sr-uuid-short>`	Average I/O latency (milliseconds).	At least one plugged VBD in SR `sr` on the host	`sr` Latency
`loadavg`	Domain0 load average. Enabled by default	None	Control Domain Load
`memory_free_kib`	Total amount of free memory (KiB). Enabled by default.	None	*Not present in XenCenter. Replaced by Used Memory.*
Not reported by the toolstack. Calculated by XenCenter.	Total amount of used memory (KiB). Enabled by default.	None	Used Memory
`memory_reclaimed`	Host memory reclaimed by squeeze (B).	None	Reclaimed Memory
`memory_reclaimed_max`	Host memory available to reclaim with squeeze (B).	None	Potential Reclaimed Memory
`memory_total_kib`	Total amount of memory (KiB) in the host. Enabled by default.	None	Total Memory
`network/latency`	Interval in seconds between the last two heartbeats transmitted from the local host to all online hosts. Disabled by default.	HA Enabled	Network Latency

Metric Name	Description	Condition	XenCenter Name
statefile/<vdi_uuid>/latency	Turn-around time in seconds of the latest State-File access from the local host. Disabled by default.	HA Enabled	HA State File Latency
pif_<pif>_rx	Bytes per second received on physical interface pif. Enabled by default.	PIF exists	XenCenter-pifname Receive (see note)
pif_<pif>_tx	Bytes per second sent on physical interface pif. Enabled by default.	PIF exists	XenCenter-pifname Send (see note)
pif_<pif>_rx_errors	Receive errors per second on physical interface pif. Disabled by default.	PIF exists	XenCenter-pifname Receive Errors (see note)
pif_<pif>_tx_errors	Transmit errors per second on physical interface pif . Disabled by default	PIF exists	XenCenter-pifname Send Errors (see note)
pif_aggr_rx	Bytes per second received on all physical interfaces. Enabled by default.	None	Total NIC Receive
pif_aggr_tx	Bytes per second sent on all physical interfaces. Enabled by default.	None	Total NIC Send
pvsaccelerator_evicted	Bytes per second evicted from the cache	PVSAccelerator Enabled	PVS-Accelerator eviction rate
pvsaccelerator_read_hits	Reads per second served from the cache	PVSAccelerator Enabled	PVS-Accelerator hit rate
pvsaccelerator_read_misses	Reads per second that cannot be served from the cache	PVSAccelerator Enabled	PVS-Accelerator miss rate

Metric Name	Description	Condition	XenCenter Name
pvsaccelerator_traffic_client_sent	Bytes per second sent by cached PVS clients	PVSAccelerator Enabled	PVS-Accelerator observed network traffic from clients
pvsaccelerator_traffic_server_sent	Bytes per second sent by cached PVS servers	PVSAccelerator Enabled	PVS-Accelerator observed network traffic from servers
pvsaccelerator_read_total	Reads per second observed by the cache	PVSAccelerator Enabled	PVS-Accelerator observed read rate
pvsaccelerator_traffic_proxy_saved	Bytes per second sent by PVSAccelerator instead of the PVS server	PVSAccelerator Enabled	PVS-Accelerator saved network traffic
pvsaccelerator_space_utilization	Percentage of space used by PVSAccelerator on this host, compared to the total size of the cache storage	PVSAccelerator Enabled	PVS-Accelerator space utilization
running_vcpus	The total number of running vCPUs	None	Number of running vCPUs
running_domains	The total number of running domains including dom0 (the control domain of the host)	None	Number of running domains
sr_<sr>_cache_size	Size in bytes of the IntelliCache SR. Enabled by default.	IntelliCache Enabled	IntelliCache Cache Size
sr_<sr>_cache_hits	Cache hits per second. Enabled by default.	IntelliCache Enabled	IntelliCache Cache Hits
sr_<sr>_cache_misses	Cache misses per second. Enabled by default.	IntelliCache Enabled	IntelliCache Cache Misses
xapi_allocation_kib	Memory (KiB) allocation done by the XAPI daemon. Enabled by default.	None	Agent Memory Allocation

Metric Name	Description	Condition	XenCenter Name
`xapi_free_memory_kib`	Free memory (KiB) available to the XAPI daemon. Enabled by default.	None	Agent Memory Free
`xapi_healthcheck/latency`	Turn-around time in seconds of the latest XAPI status monitoring call on the local host. Disabled by default.	High availability Enabled	XenServer High Availability Latency
`xapi_live_memory_kib`	Live memory (KiB) used by XAPI daemon. Enabled by default.	None	Agent Memory Live
`xapi_memory_usage_kib`	Total memory (KiB) allocated used by XAPI daemon. Enabled by default.	None	Agent Memory Usage

Available VM metrics

Metric Name	Description	Condition	XenCenter Name
`cpu<cpu>`	Utilization of vCPU `cpu` (fraction). Enabled by default	vCPU `cpu` exists	CPU
`cpu_usage`	Domain CPU usage	None	cpu_usage
`memory`	Memory currently allocated to VM (Bytes).Enabled by default	None	Total Memory
`memory_target`	Target of VM balloon driver (Bytes). Enabled by default	None	Memory target
`memory_internal_free`	Memory used as reported by the guest agent (KiB). Enabled by default	None	Free Memory

Metric Name	Description	Condition	XenCenter Name
`runstate_fullrun`	Fraction of time that all vCPUs are running.	None	vCPUs full run
`runstate_full_contention`	Fraction of time that all vCPUs are runnable (that is, waiting for CPU)	None	vCPUs full contention
`runstate_concurrency_hazard`	Fraction of time that some vCPUs are running and some are runnable	None	vCPUs concurrency hazard
`runstate_blocked`	Fraction of time that all vCPUs are blocked or offline	None	vCPUs idle
`runstate_partial_run`	Fraction of time that some vCPUs are running, and some are blocked	None	vCPUs partial run
`runstate_partial_contention`	Fraction of time that some vCPUs are runnable and some are blocked	None	vCPUs partial contention
`vbd_<vbd>_write`	Writes to device `vbd` in bytes per second. Enabled by default	VBD `vbd` exists	Disk `vbd` Write
`vbd_<vbd>_read`	Reads from device `vbd` in bytes per second. Enabled by default.	VBD `vbd` exists	Disk `vbd` Read
`vbd_<vbd>_write_latency`	Writes to device `vbd` in microseconds.	VBD `vbd` exists	Disk `vbd` Write Latency
`vbd_<vbd>_read_latency`	Reads from device `vbd` in microseconds.	VBD `vbd` exists	Disk `vbd` Read Latency
`vbd <vbd>_iops_read`	Read requests per second.	At least one plugged VBD for non-ISO VDI on the host	Disk `vbd` Read IOPs
`vbd <vbd>_iops_write`	Write requests per second.	At least one plugged VBD for non-ISO VDI on the host	Disk `vbd` Write IOPS

Metric Name	Description	Condition	XenCenter Name
vbd <vbd> _iops_total	I/O requests per second.	At least one plugged VBD for non-ISO VDI on the host	Disk vbd Total IOPS
vbd <vbd> _iowait	Percentage of time waiting for I/0.	At least one plugged VBD for non-ISO VDI on the host	Disk vbd IO Wait
vbd <vbd> _inflight	Number of I/O requests currently in flight.	At least one plugged VBD for non-ISO VDI on the host	Disk vbd Inflight Requests
vbd <vbd> _avgqu_sz	Average I/O queue size.	At least one plugged VBD for non-ISO VDI on the host	Disk vbd Queue Size
vif_<vif>_rx	Bytes per second received on virtual interface number vif. Enabled by default.	VIF vif exists	vif Receive
vif_<vif>_tx	Bytes per second transmitted on virtual interface vif. Enabled by default.	VIF vif exists	vif Send
vif_<vif> _rx_errors	Receive errors per second on virtual interface vif. Enabled by default.	VIF vif exists	vif Receive Errors
vif_<vif> _tx_errors	Transmit errors per second on virtual interface vif Enabled by default.	VIF vif exists	vif Send Errors

Note:

The value of <XenCenter-pif-name> can be any of the following:

- NIC <pif> - if <pif> contains pif_eth#, where ## is 0–9
- <pif> - if <pif> contains pif_eth#.## or pif_xenbr## or pif_bond##
- <Internal> Network <pif> - if <pif> contains pif_xapi##, (note that < Internal> appears as is)
- TAP <tap> - if <pif> contains pif_tap##

> - `xapi Loopback` - if `<pif>` contains `pif_lo`

Analyze and visualize metrics

The Performance tab in XenCenter provides real time monitoring of performance statistics across resource pools in addition to graphical trending of virtual and physical machine performance. Graphs showing CPU, memory, network, and disk I/O are included on the Performance tab by default. You can add more metrics, change the appearance of the existing graphs or create extra ones. For more information, see *Configuring metrics* in the following section.

- You can view up to 12 months of performance data and zoom in to take a closer look at activity spikes.
- XenCenter can generate performance alerts when CPU, memory, network I/O, storage I/O, or disk I/O usage exceed a specified threshold on a host, VM, or SR. For more information, see *Alerts* in the following section.

> **Note:**
>
> Install the XenServer VM Tools to see full VM performance data.

Configure performance graphs **To add a graph:**

1. On the **Performance** tab, click **Actions** and then **New Graph**. The New Graph dialog box is displayed.
2. In the **Name** field, enter a name for the graph.
3. From the list of **Datasources**, select the check boxes for the datasources you want to include in the graph.
4. Click **Save**.

To edit an existing graph:

1. Navigate to the **Performance** tab, and select the graph that you would like to modify.
2. Right-click on the graph and select **Actions**, or click the **Actions** button. Then select **Edit Graph**.
3. On the graph details window, make the necessary changes, and click **OK**.

Configure the graph type Data on the performance graphs can be displayed as lines or as areas. To change the graph type:

1. On the **Tools** menu, click **Options** and select **Graphs**.

2. To view performance data as a line graph, click the **Line graph** option.
3. To view performance data as an area graph, click the **Area graph** option.
4. Click **OK** to save your changes.

Comprehensive details for configuring and viewing XenCenter performance graphs can be found in the XenCenter documentation in the section Monitoring System Performance.

Configure metrics

> **Note:**
>
> C-states and P-states are power management features of some processors. The range of states available depends on the physical capabilities of the host, as well power management configuration.

Both host and VM commands return the following:

- A full description of the data source
- The units applied to the metric
- The range of possible values that may be used

For example:

```
    name_label: cpu0-C1
    name_description: Proportion of time CPU 0 spent in C-state 1
    enabled: true
    standard: true
    min: 0.000
    max: 1.000
    units: Percent
<!--NeedCopy-->
```

Enable a specific metric Most metrics are enabled and collected by default, to enable those metrics that are not, enter the following:

```
xe host-data-source-record data-source=metric name host=hostname
<!--NeedCopy-->
```

Disable a specific metric You might not want to collect certain metrics regularly. To disable a previously enabled metric, enter the following:

```
xe host-data-source-forget data-source=metric name host=hostname
<!--NeedCopy-->
```

Display a list of currently enabled host metrics To list the host metrics currently being collected, enter the following:

```
xe host-data-source-list host=hostname
<!--NeedCopy-->
```

Display a list of currently enabled VM metrics To host the VM metrics currently being collected, enter the following:

```
xe vm-data-source-list vm=vm_name
<!--NeedCopy-->
```

Use RRDs

XenServer uses RRDs to store performance metrics. These RRDs consist of multiple Round Robin Archives (RRAs) in a fixed size database.

Each archive in the database samples its particular metric on a specified granularity:

- Every 5 seconds for 10 minutes
- Every minute for the past two hours
- Every hour for the past week
- Every day for the past year

The sampling that takes place every five seconds records actual data points, however the following RRAs use Consolidation Functions instead. The consolidation functions supported by XenServer are:

- AVERAGE
- MIN
- MAX

RRDs exist for individual VMs (including dom0) and the XenServer host. VM RRDs are stored on the host on which they run, or the pool coordinator when not running. Therefore the location of a VM must be known to retrieve the associated performance data.

For detailed information on how to use XenServer RRDs, see the XenServer Software Development Kit Guide.

Analyze RRDs using HTTP

You can download RRDs over HTTP from the XenServer host specified using the HTTP handler registered at /`host_rrd` or /`vm_rrd`. Both addresses require authentication either by HTTP authentication, or by providing a valid management API session references as a query argument. For example:

Download a Host RRD.

```
wget http://server/host_rrd?session_id=OpaqueRef:SESSION HANDLE>
<!--NeedCopy-->
```

Download a VM RRD.

```
wget http://server/vm_rrd?session_id=OpaqueRef:SESSION HANDLE>&uuid=VM
    UUID>
<!--NeedCopy-->
```

Both of these calls download XML in a format that can be imported into the `rrdtool` for analysis, or parsed directly.

Analyze RRDs using rrd2csv

In addition to viewing performance metrics in XenCenter, the rrd2csv tool logs RRDs to Comma Separated Value (CSV) format. Man and help pages are provided. To display the rrd2csv tool man or help pages, run the following command:

```
man rrd2csv
<!--NeedCopy-->
```

Or

```
rrd2csv --help
<!--NeedCopy-->
```

> **Note:**
>
> Where multiple options are used, supply them individually. For example: to return both the UUID and the name-label associated with a VM or a host, call rrd2csv as shown below:
>
> `rrd2csv -u -n`
>
> The UUID returned is unique and suitable as a primary key, however the name-label of an entity might not necessarily be unique.

The man page (`rrd2csv --help`) is the definitive help text of the tool.

Alerts

You can configure XenServer to generate alerts based on any of the available Host or VM Metrics. In addition, XenServer provides preconfigured alerts that trigger when hosts undergo certain conditions and states. You can view these alerts using XenCenter or the xe CLI.

View alerts using XenCenter

You can view different types of alerts in XenCenter by clicking **Notifications** and then **Alerts**. The Alerts view displays various types of alerts, including Performance alerts, System alerts, and Software update alerts.

Performance alerts

Performance alerts can be generated when one of the following values exceeds a specified threshold on a managed host, VM, or storage repository (SR): CPU usage, network usage, memory usage, control domain memory usage, storage throughput, or VM disk usage.

By default, the alert repeat interval is set to 60 minutes, it can be modified if necessary. Alerts are displayed on the Alerts page in the Notifications area in XenCenter. You can also configure XenCenter to send an email for any specified performance alerts along with other serious system alerts.

Any customized alerts that are configured using the xe CLI are also displayed on the Alerts page in XenCenter.

Each alert has a corresponding priority/severity level. You can modify these levels and optionally choose to receive an email when the alert is triggered. The default alert priority/severity is set at 3 .

Priority	Name	Description	Default Email Alert
1	Critical	Act now or data may be permanently lost/corrupted.	Yes
2	Major	Act now or some services may fail.	Yes
3	Warning	Act now or a service may suffer.	Yes
4	Minor	Notice that something just improved.	No
5	Information	Day-to-day information (VM Start, Stop, Resume and so on)	No
?	Unknown	Unknown error	No

Configure performance alerts

1. In the **Resources** pane, select the relevant host, VM, or SR, then click the **General** tab and then **Properties**.
2. Select the **Alerts** tab. The following table summarizes which alerts are available for hosts, VMs, or SRs:

Alert name	Host	VM	SR	Description
Generate CPU usage alerts	X	X		Set the CPU usage and time threshold that trigger the alert.
Generate control domain CPU usage alerts	X			Set the control domain CPU usage and time threshold that trigger the alert.
Generate memory usage alerts	X			Set the memory usage and time threshold that trigger the alert.
Generate control domain memory usage alerts	X			Set the control domain memory usage and time threshold that trigger the alert.
Generate control domain free memory alerts	X			Set the control domain free memory and time threshold that trigger the alert.
Generate disk usage alerts		X		Set the disk usage and time threshold trigger the alert.

Alert name	Host	VM	SR	Description
Generate storage throughput alerts			X	Set the storage throughput and time threshold that trigger the alert. Note: Physical Block Devices (PBD) represent the interface between a specific XenServer host and an attached SR. When the total read/write SR throughput activity on a PBD exceeds the threshold you have specified, alerts are generated on the host connected to the PBD. Unlike other XenServer host alerts, this alert must be configured on the SR.
Generate network usage alerts	X	X		Set the network usage and time threshold that trigger the alert.

To change the alert repeat interval, enter the number of minutes in the **Alert repeat interval** box. When an alert threshold has been reached and an alert generated, another alert is not generated until after the alert repeat interval has elapsed.

3. Click **OK** to save your changes.

For comprehensive details on how to view, filter and configure severities for performance alerts, see Configuring Performance Alerts in the XenCenter documentation.

System alerts

The following table displays the system events/conditions that trigger an alert to be displayed on the Alerts page in XenCenter.

Name	Priority/Severity	Description
license_expires_soon	2	XenServer License agreement expires soon.
ha-statefile_lost	2	Lost contact with the high availability Storage Repository, act soon.
ha-heartbeat_approaching_timeout	5	High availability approaching timeout, host may reboot unless action is taken.
ha_statefile_approaching_timeout	5	High availability approaching timeout, host may reboot unless action is taken.
haxapi_healthcheck_approaching_timeout	5	High availability approaching timeout, host may reboot unless action is taken.
ha_network_bonding_error	3	Potential service loss. Loss of network that sends high availability heartbeat.
ha_pool_overcommited	3	Potential service loss. High availability is unable to guarantee protection for configured VMs.
ha_poor_drop_in_plan_exists_for	3	High availability coverage has dropped, more likely to fail, no loss present yet.
ha_protected_vm_restart_failed	2	Service Loss. High availability was unable to restart a protected VM.

Name	Priority/Severity	Description
ha_host_failed	3	High availability detected that a host failed.
ha_host_was_fenced	4	High availability rebooted a host to protect against VM corruption.
redo_log_healthy	4	The XAPI redo log has recovered from a previous error.
redo_log_broken	3	The XAPI redo log has encountered an error.
ip_configured_pif_can_unplug	3	An IP configured NIC can be unplugged by XAPI when using high availability, possibly leading to high availability failure.
host_sync_data_failed	3	Failed to synchronize XenServer performance statistics.
host_clock_skew_detected	3	The host clock is not synchronized with other hosts in the pool.
host_clock_went_backwards	1	The host clock is corrupted.
pool_master_transition	4	A new host has been specified as pool coordinator.
pbd_plug_failed_on_server_start	3	The host failed to connect to Storage at boot time.
auth_external_init_failed	2	The host failed to enable external AD authentication.
auth_external_pool_non-homogeneous	2	Hosts in a pool have different AD authentication configuration.
multipath_period_alert	3	A path to an SR has failed or recovered.
bond-status-changed	3	A link in a bond has disconnected or reconnected.

Software update alerts

- **XenCenter old:** 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:** 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

Configure performance alerts by using the xe CLI

> **Note:**
>
> Triggers for alerts are checked at a minimum interval of five minutes. This interval avoids placing excessive load on the system to check for these conditions and reporting of false positives. Setting an alert repeat interval smaller than five minutes results in the alerts still being generated at the five minute minimum interval.

The performance monitoring `perfmon` tool runs once every five minutes and requests updates from XenServer which are averages over one minute. These defaults can be changed in `/etc/sysconfig/perfmon`.

The `perfmon` tool reads updates every five minutes of performance variables 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 host, `perfmon` reads the parameter `other-config:perfmon` and uses this string to determine which variables to monitor, and under which circumstances to generate a message.

For example, the following shows an example of configuring a VM "CPU usage"alert by writing an XML string into the parameter `other-config:perfmon`:

```
xe vm-param-set uuid=vm_uuid other-config:perfmon=\

'<config>
    <variable>
        <name value="cpu_usage"/>
        <alarm_trigger_level value="0.5"/>
    </variable>
</config>'
<!--NeedCopy-->
```

> **Note:**
>
> You can use multiple variable nodes.

After setting the new configuration, use the following command to refresh `perfmon` for each host:

```
xe host-call-plugin host=host_uuid plugin=perfmon fn=refresh
<!--NeedCopy-->
```

If this refresh is not done, there is a delay before the new configuration takes effect, since by default, `perfmon` checks for new configuration every 30 minutes. This default can be changed in `/etc/sysconfig/perfmon`.

Valid VM elements

- `name`: The name of the variable (no default). If the name value is either `cpu_usage`, `network_usage`, or `disk_usage`, the `rrd_regex` and `alarm_trigger_sense` parameters are not required as defaults for these values are used.
- `alarm_priority`: The priority of the alerts generated (default 3).
- `alarm_trigger_level`: The level of value that triggers an alert (no default).
- `alarm_trigger_sense`: The value is `high` if `alarm_trigger_level` is a maximum value otherwise `low` if the `alarm_trigger_level` is a minimum value (the default `high`).
- `alarm_trigger_period`: The number of seconds that values (above or below the alert threshold) can be received before an alert is sent (the default is `60`).
- `alarm_auto_inhibit_period`: The number of seconds this alert will be disabled after an alert is sent (the default is `3600`).
- `consolidation_fn`: Combines variables from rrd_updates into one value. For `cpu-usage` the default is `average`, for `fs_usage` the default is`get_percent_fs_usage` and for all others - `sum`.
- `rrd_regex`: Matches the names of variables from `xe vm-data-sources-list uuid=vm_uuid`, to compute performance values. This parameter has defaults for the named variables:
  - cpu_usage
  - memory_internal_free
  - network_usage
  - disk_usage

If specified, the values of all items returned by `xe vm-data-source-list` whose names match the specified regular expression are consolidated using the method specified as the `consolidation_fn`.

Valid host elements

- `name`: The name of the variable (no default).
- `alarm_priority`: The priority of the alerts generated (default 3).
- `alarm_trigger_level`: The level of value that triggers an alert (no default).
- `alarm_trigger_sense`: The value is `high` when `alarm_trigger_level` is a maximum value otherwise `low` if the `alarm_trigger_level` is a minimum value. (default `high`)
- `alarm_trigger_period`: The number of seconds that values (above or below the alert threshold) can be received before an alert is sent (default `60`).
- `alarm_auto_inhibit_period`: The number of seconds that the alert is disabled for after an alert is sent. (default `3600`).
- `consolidation_fn`: Combines variables from `rrd_updates` into one value (default `sum` - or `average`)
- `rrd_regex`: A regular expression to match the names of variables returned by the `xe vm -data-source-list uuid=vm_uuid` command to use to compute the statistical value. This parameter has defaults for the following named variables:
  - cpu_usage
  - network_usage
  - memory_free_kib
  - sr_io_throughput_total_xxxxxxxx (where `xxxxxxxx`is the first eight characters of the SR-UUID).

SR Throughput: Storage throughput alerts must be configured on the SR rather than the host. For example:

```
xe sr-param-set uuid=sr_uuid other-config:perfmon=\
'<config>
    <variable>
        <name value="sr_io_throughput_total_per_host"/>
        <alarm_trigger_level value="0.01"/>
    </variable>
</config>'
<!--NeedCopy-->
```

Generic example configuration The following example shows a generic configuration:

```
<config>
    <variable>
    <name value="NAME_CHOSEN_BY_USER"/>
    <alarm_trigger_level value="THRESHOLD_LEVEL_FOR_ALERT"/>
    <alarm_trigger_period value="
       RAISE_ALERT_AFTER_THIS_MANY_SECONDS_OF_BAD_VALUES"/>
    <alarm_priority value="PRIORITY_LEVEL"/>
    <alarm_trigger_sense value="HIGH_OR_LOW"/>
    <alarm_auto_inhibit_period value="
       MINIMUM_TIME_BETWEEN_ALERT_FROM_THIS_MONITOR"/>
    <consolidation_fn value="FUNCTION_FOR_COMBINING_VALUES"/>
    <rrd_regex value="REGULAR_EXPRESSION_TO_CHOOSE_DATASOURCE_METRIC"/>
    </variable>

    <variable>
    ...
    </variable>

    ...
</config>
<!--NeedCopy-->
```

Configure email alerts

You can configure XenServer to send email notifications when XenServer hosts generate alerts. The mail-alarm utility in XenServer uses sSMTP to send these email notifications. You can enable basic email alerts by using XenCenter or the xe Command Line Interface (CLI). For further configuration of email alerts, you can modify the `mail-alarm.conf` configuration file.

Use an SMTP server that does not require authentication. Emails sent through SMTP servers that require authentication cannot be delivered.

Enable email alerts by using XenCenter

1. In the `Resources` pane, right-click on a pool and select `Properties`.
2. In the `Properties` window, select `Email Options`.
3. Select the `Send email alert notifications` check box. Enter your preferred destination address for the notification emails and SMTP server details.
4. Choose your preferred language from the `Mail language` list. The default language for performance alert emails is English.

Enable email alerts by using the xe CLI

To configure email alerts, specify your preferred destination address for the notification emails and SMTP server:

```
xe pool-param-set uuid=pool_uuid other-config:mail-destination=joe.
    bloggs@example.com
xe pool-param-set uuid=pool_uuid other-config:ssmtp-mailhub=smtp.
    example.com:<port>
<!--NeedCopy-->
```

XenServer automatically configures the sender address as `noreply@<hostname>`. However, you can set the sender address explicitly:

```
xe pool-param-set uuid=pool_uuid other-config:mail-sender=
    serveralerts@example.com
<!--NeedCopy-->
```

When you turn on email notifications, you receive an email notification when an alert with a priority of 3 or higher is generated. Therefore, the default minimum priority level is 3. You can change this default with the following command:

```
xe pool-param-set uuid=pool_uuid other-config:mail-min-priority=level
<!--NeedCopy-->
```

> **Note:**
>
> Some SMTP servers only forward mails with addresses that use FQDNs. If you find that emails are not being forwarded it might be for this reason. In which case, you can set the server host name to the FQDN so this address is used when connecting to your mail server.

To configure the language for the performance alert emails:

```
xe pool-param-set uuid=pool_uuid other-config:mail-language=ja-JP
<!--NeedCopy-->
```

The default language for performance alert emails is English.

Further configuration

To further configure the mail-alarm utility in XenServer, create an `/etc/mail-alarm.conf` file containing the following:

```
root=postmaster
authUser=<username>
authPass=<password>
mailhub=@MAILHUB@
<!--NeedCopy-->
```

`/etc/mail-alarm.conf` is a user-supplied template for sSMTP's configuration file `ssmtp.conf` and is used for all alerts generated by XenServer hosts. It consists of keys where `key=@KEY@` and `@KEY@` is replaced by the corresponding value of `ssmtp-key` in `pool.other_config`. These values are then passed to ssmtp, allowing you to control aspects of the sSMTP configuration using values from `pool.other_config`. Note how `@KEY@` (uppercase) corresponds to `ssmtp-key` (lowercase, prefixed by `ssmtp-`).

For example, if you set the SMTP server:

```
xe pool-param-set uuid=pool_uuid other-config:ssmtp-mailhub=smtp.
    example.com
<!--NeedCopy-->
```

and then add the following to your `/etc/mail-alarm.conf` file:

```
mailhub=@MAILHUB@
<!--NeedCopy-->
```

`mailhub=@MAILHUB@` becomes `mailhub=smtp.example.com`.

Each SMTP server can differ slightly in its setup and may require extra configuration. To further configure sSMTP, modify its configuration file `ssmtp.conf`. By storing relevant keys in the `mail-alarm.conf` file, you can use the values in `pool.other_config` to configure sSMTP. The following extract from the `ssmtp.conf` man page shows the correct syntax and available options:

```
NAME
    ssmtp.conf – ssmtp configuration file

DESCRIPTION
    ssmtp reads configuration data from /etc/ssmtp/ssmtp.conf The file
       con-
    tains keyword-argument pairs, one per line. Lines starting with '#'
    and empty lines are interpreted as comments.

The possible keywords and their meanings are as follows (both are case-
insensitive):

    Root
    The user that gets all mail for userids less than 1000. If blank,
    address rewriting is disabled.

    Mailhub
        The host to send mail to, in the form host | IP_addr port :
        <port>. The default port is 25.

    RewriteDomain
    The domain from which mail seems to come. For user authentication.

    Hostname
        The full qualified name of the host. If not specified, the host
        is queried for its hostname.

    FromLineOverride
        Specifies whether the From header of an email, if any, may over
            -
        ride the default domain. The default is "no".

    UseTLS
    Specifies whether ssmtp uses TLS to talk to the SMTP server.
    The default is "no".

    UseSTARTTLS
        Specifies whether ssmtp does a EHLO/STARTTLS before starting
            TLS
        negotiation. See RFC 2487.

    TLSCert
        The file name of an RSA certificate to use for TLS, if required
            .

    AuthUser
        The user name to use for SMTP AUTH. The default is blank, in
        which case SMTP AUTH is not used.

    AuthPass
        The password to use for SMTP AUTH.

    AuthMethod
        The authorization method to use. If unset, plain text is used.
        May also be set to "cram-md5".
<!--NeedCopy-->
```

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. For more information, see Monitoring System Performance.

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. For more information, see Monitoring System Performance.

Determine throughput of physical bus adapters

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

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.

Monitor host and dom0 resources with NRPE

> **Note:**
>
> The NRPE feature is available for XenServer Premium or Trial Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.

Users with the Pool Admin role can use any third-party monitoring tool that supports the Nagios Remote Plugin Executor (NRPE) to monitor resources consumed by your XenServer host and dom0 - the control domain of your host.

You can use the following check plugins to monitor host and dom0 resources:

Metric	NRPE check name	Description	Default warning threshold	Default critical threshold	Performance data returned
Host CPU Load	check_host_load	Gets and checks the current load per physical CPU of the host, where load refers to the number of vCPU(s) in a running or runnable state.	3	4	Current system load of the CPU of the host (calculated by taking the average load of the physical CPU of the host).
Host CPU Usage (%)	check_host_cpu	Gets and checks the current average overall CPU usage of the host.	80%	90%	The percentage of host CPU that is currently free and the percentage that is in use.

Metric	NRPE check name	Description	Default warning threshold	Default critical threshold	Performance data returned
Host Memory Usage (%)	check_host_memory	Gets and checks the current memory usage of the host.	80%	90%	The percentage of host memory that is currently free and the percentage that is in use.
Host vGPU Usage (%)	check_vgpu	Gets and checks all the current running Nvidia vGPU usage of the host.	80%	90%	The percentage of running vGPU that is currently free and the percentage that is in use.
Host vGPU Memory Usage (%)	check_vgpu_memory	Gets and checks all the current running Nvidia vGPU memory usage (including the shared memory and graphic memory) of the host.	80%	90%	The percentage of running vGPU memory (including the shared memory and graphic memory) that is currently free and the percentage that is in use.

Metric	NRPE check name	Description	Default warning threshold	Default critical threshold	Performance data returned
Dom0 CPU Load	check_load	Gets and checks the current system load average per CPU of dom0, where load refers to the number of processes in a running or runnable state.	2.7,2.6,2.5	3.2,3.1,3	Host CPU load data calculated by taking the average of the last 1, 5, and 15 minutes.
Dom0 CPU Usage (%)	check_cpu	Gets and checks the current average overall CPU usage of dom0.	80%	90%	The average overall CPU usage of dom0 as a percentage.
Dom0 Memory Usage (%)	check_memory	Gets and checks the current memory usage of dom0.	80%	90%	The percentage of dom0 memory that is currently free and the percentage that is in use.
Dom0 Free Swap (%)	check_swap	Gets and checks the current swap usage of dom0.	20%	10%	The percentage of MB on dom0 that is currently free.

Metric	NRPE check name	Description	Default warning threshold	Default critical threshold	Performance data returned
Dom0 Root Partition Free Space (%)	check_disk_root	Gets and checks the current root partition usage of dom0.	20%	10%	The percentage of MB on the dom0 root partition that is currently free.
Dom0 Log Partition Free Space (%)	check_disk_log	Gets and checks the current log partition usage of dom0.	20%	10%	The percentage of MB on the dom0 log partition that is currently free.
Toolstack Status	check_xapi	Gets and checks the status of the XenServer management toolstack (also known as XAPI).			XAPI elapsed uptime in seconds.

Metric	NRPE check name	Description	Default warning threshold	Default critical threshold	Performance data returned
Multipath Status	check_multipath	Gets and checks the status of the storage paths.			The status of the storage paths. `OK` indicates that all paths are active, `WARNING` indicates that some paths have failed but more than one path is active, `CRITICAL` indicates that there is only one path active or that all paths have failed, `UNKNOWN` indicates that host multipathing is disabled and that the status of the paths cannot be fetched.

NRPE is an on-premises service that runs in dom0 and listens on TCP port (default) 5666 for check execution requests from a monitoring tool. After a request arrives, NRPE parses it, finds the corresponding check command including the parameter's details from the configuration file, and then runs it. The result of the check is sent to the monitoring tool, which stores the results of past checks and provides a graph showing the historical performance data.

Prerequisites

To be able to use NRPE to monitor host and dom0 resources, the monitoring tool you are using must meet the following prerequisites:

- The monitoring tool must be compatible with NRPE version 4.1.0.
- To allow communication between NRPE and the monitoring tool, the monitoring tool must support TLS 1.2 with ciphers `ECDHE-RSA-AES256-GCM-SHA384` and `ECDHE-RSA-AES128-GCM-SHA256`, and the EC curve is `secp384r1`.

Constraints

- You can configure NRPE settings for an entire pool or for a standalone host that is not part of a pool. Currently, you cannot configure NRPE settings for an individual host in a pool.
- If you add a host to a pool that already has NRPE enabled and configured on it, XenCenter does not automatically apply the pool's NRPE settings to the new host. You must reconfigure NRPE settings on the pool after adding the new host or configure the new host with same NRPE settings before adding it to the pool.

  > **Note:**
  >
  > When reconfiguring NRPE settings on a pool after adding a new host, ensure the host is up and running.

- If a host is removed from a pool with NRPE enabled and configured on it, XenCenter does not alter the NPRE settings on the host or the pool.

Configure NRPE by using the xe CLI

You can configure NRPE by using the xe CLI or XenCenter. For more information on how to configure NRPE by using XenCenter, see Monitoring host and dom0 resources with NRPE.

After making configuration changes to NRPE, restart the NRPE service by using:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=restart
<!--NeedCopy-->
```

Enable NRPE NRPE is disabled by default in XenServer. To enable NRPE on a host's control domain (dom0), run the following commands in the xe CLI:

1. Get the host UUID of the host that you want to monitor:

   `xe host-list`

2. Enable NRPE on the host:

   `xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=enable`

   If the operation runs successfully, this command outputs `Success`. When XenServer restarts, NRPE starts automatically.

To stop, start, restart, or disable NRPE:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=<operation>
<!--NeedCopy-->
```

where **operation** is `stop`, `start`, `restart`, or `disable`.

Monitoring servers This is a comma-delimited list of IP addresses or host names that are allowed to talk to the NRPE daemon. Network addresses with a bit mask (for example `192.168.1.0/24`) are also supported.

View the current list of monitoring servers:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=get-config
    args:allowed_hosts
<!--NeedCopy-->
```

Allow the monitoring tool to execute checks:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=set-config
    args:allowed_hosts=<IP address or hostname>
<!--NeedCopy-->
```

Query all NRPE settings:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=get-config
<!--NeedCopy-->
```

Configure multiple NRPE settings:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=set-config
    args:allowed_hosts=<IP address or hostname> args:ssl_logging=<SSL
    log level> args:debug=<debug log level>
<!--NeedCopy-->
```

Logs

Debug logging By default, debug logging is disabled.

To check whether debug logging is enabled, run the following command:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=get-config
    args:debug
<!--NeedCopy-->
```

If `debug: 0` is returned, debug logging is disabled.

To enable debug logging:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=set-config
    args:debug=1
<!--NeedCopy-->
```

SSL logging By default, SSL logging is disabled:

```
ssl_logging=0x00
<!--NeedCopy-->
```

To check whether SSL logging is enabled, run the following command:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=get-config
    args:ssl_logging
<!--NeedCopy-->
```

To enable SSL logging:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=set-config
    args:ssl_logging=0x2f
<!--NeedCopy-->
```

Warning and critical thresholds For some of these check plugins, you can set warning and critical threshold values so that if the value returned by a check plugin exceeds the threshold values, an alert is generated. The warning threshold indicates a potential issue and the critical threshold indicates a more serious issue that requires immediate attention. Although default values are set for the warning and critical thresholds, you can adjust the threshold values.

To query the default warning and critical threshold values for all the checks, run the following xe CLI command which returns a list of all the checks and their associated warning and critical thresholds:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=get-threshold
<!--NeedCopy-->
```

You can also query the threshold values for a specific check. For example, to get the warning and critical threshold values for the `check_memory` check plugin, run the following xe CLI command:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=get-threshold
    args:check_memory
<!--NeedCopy-->
```

You can also change the default value of a threshold. For example, to change the default threshold values for the `check_memory` check plugin, run the following xe CLI command:

```
xe host-call-plugin host-uuid=<host uuid> plugin=nrpe fn=set-threshold
    args:check_memory args:w=75 args:c=85
<!--NeedCopy-->
```

Monitor host and dom0 resources with SNMP

> **Note:**
>
> The SNMP feature is available for XenServer Premium or Trial Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.

With the Pool Admin role, you can use SNMP to remotely monitor resources consumed by your XenServer host and dom0 - the control domain of your host. An SNMP manager, also known as a network management system (NMS), sends query requests to an SNMP agent running on a XenServer host. The SNMP agent replies to these query requests by sending data collected on various metrics back to the NMS. The data that can be collected is defined by object identifiers (OIDs) in a text file called a management information base (MIB). An OID represents a specific piece of measurable information about a network device, such as CPU or memory usage.

You can also configure traps, which are agent-initiated messages that alert the NMS that a specific event has occurred in XenServer. Both query requests and traps can be used to monitor the status of your XenServer pools. These are defined as metric and trap objects and are identified by OIDs in a MIB file `XENSERVER-MIB.txt`, available to download from the XenServer Downloads page. The following tables provide information about these metric and trap objects.

Metric objects

You can request a specific piece of information about your XenServer hosts by using the metrics listed in the following table. These metrics are used by the SNMP manager when sending query requests to an SNMP agent and so you can view this data in your NMS.

You can view the returned data from these metric objects from your NMS or from the xe CLI. To query the metric objects from the xe CLI, run `host-data-source-query` or `vm-data-source-query` and provide the RRDD data source as a value for the `data-source` parameter. For example:

```
xe host-data-source-query data-source=cpu_avg host=<host UUID>
<!--NeedCopy-->
```

> **Note:**
>
> By default, the NMS sends OID query requests to SNMP agents using port 161.

Object identifier (OID)	RRDD data source	Returned data	Type
.1.3.6.1.4.1.60953.1.1.1.1	`memory`	Dom0 total memory in MB	Unsigned32
.1.3.6.1.4.1.60953.1.1.1.2	`memory_internal_free`	Dom0 free memory in MB	Unsigned32
.1.3.6.1.4.1.60953.1.1.1.3	`cpu_usage`	Dom0 CPU usage as a percentage	Float
.1.3.6.1.4.1.60953.1.1.1.4	`memory_total_kib`	Host total memory in MB	Unsigned32
.1.3.6.1.4.1.60953.1.1.1.5	`memory_free_kib`	Host free memory in MB	Unsigned32
.1.3.6.1.4.1.60953.1.1.1.6	`cpu_avg`	Host CPU usage as a percentage	Float
.1.3.6.1.4.1.60953.1.1.1.7	(see note 1)	pCPUs number	Unsigned32
.1.3.6.1.4.1.60953.1.1.1.8	`running_vcpus`	Running vCPUs number	Unsigned32
.1.3.6.1.4.1.60953.1.1.1.9	`running_domains`	Running VMs number	Unsigned32

> **Notes:**
>
> 1. The name of a pCPU is in the format `cpu` followed by a number. To query the number of pCPUs from the xe CLI, run the following command:
>
>    ```
>    xe host-data-source-list host=<host UUID> | grep -E 'cpu[0-9]+
>    $'
>    ```
>
>   This returns a list of the CPU metrics that match the regular expression `cpu[0-9]+`.

Traps

Traps are alerts sent by the SNMP agent to notify the SNMP manager when certain events occur, allowing you to monitor your XenServer hosts and identify issues early. You can configure your SNMP settings to generate a trap when a limit is reached (for example, if the host CPU usage is too high). When a trap is generated, it is sent to your NMS and the following fields are returned as part of the trap object.

> **Note:**
>
> By default, the SNMP agent on the pool coordinator host sends traps to the NMS using UPD port 162.

Object identifier (OID)	Field name	Type	Description
.1.3.6.1.4.1.60953.1.10.1.1	`operation`	String	Can be one of the following values: `add` or `del`. `operation` is `add` if a trap is generated by XenServer and sent to your NMS (an alert is also created in XenCenter) or `del` if an alert is destroyed (for example, if you dismiss an alert).
.1.3.6.1.4.1.60953.1.10.1.2	`ref`	String	The reference for the trap object.
.1.3.6.1.4.1.60953.1.10.1.3	`uuid`	String	The UUID of the trap object.
.1.3.6.1.4.1.60953.1.10.1.4	`name`	String	The name of the trap object.
.1.3.6.1.4.1.60953.1.10.1.5	`priority`	Integer	The severity of the trap. Can be one of the following values: 1: Critical, 2: Major, 3: Warning, 4: Minor, 5: Information, `others`: Unknown.
.1.3.6.1.4.1.60953.1.10.1.6	**`class`**	String	The category of the trap generated. Can be one of the following values: `VM`, `Host`, `SR`, `Pool`, `VMPP`, `VMSS`, `PVS_proxy`, `VDI`, or `Certificate`.

Object identifier (OID)	Field name	Type	Description
.1.3.6.1.4.1.60953.1.10.1.7	`obj-uuid`	String	The xapi object UUID of the various classes of the field **`class`**.
.1.3.6.1.4.1.60953.1.10.1.8	`timestamp`	String	The time at which the trap is generated.
.1.3.6.1.4.1.60953.1.10.1.9	`body`	String	Detailed information about the field `name`.

Prerequisites

- All hosts in a pool must be running the same XenServer version and this version must contain the SNMP plugin.

  > **Note:**
  >
  > If you cannot see the **SNMP** tab in XenCenter, it might be because the host or a member of the pool is not running a version of XenServer that supports SNMP. Update the host or pool to the latest version of XenServer.

- The NMS you are using must support SNMPv2c or SNMPv3.
- Your NMS and XenServer must be network-connected.

Constraints

- You can configure SNMP settings for an entire pool or for a standalone host that is not part of a pool. Currently, you cannot configure SNMP settings for an individual host in a pool.
- If you add a host to a pool that already has SNMP enabled and configured on it, XenCenter does not automatically apply the pool's SNMP settings to the new host. You must reconfigure SNMP settings on the pool after adding the new host or configure the new host with the same SNMP settings before adding it to the pool.

  > **Note:**
  >
  > When reconfiguring SNMP settings on a pool after adding a new host, ensure the host is up and running and not in maintenance mode.

- Before performing a rolling pool upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8 or applying updates to your XenServer hosts and pools, back up the following configuration files if you manually modified them before and need them:

  - `/etc/snmp/snmpd.xs.conf`
  - `/etc/sysconfig/snmp`
  - `/var/lib/net-snmp/snmpd.conf`

- When the SNMP agent is offline, traps cannot be generated. For example, if the SNMP agent is restarted or the pool coordinator is rebooted or re-designated.

Configure SNMP by using the xe CLI

You can configure SNMP by using the xe CLI or XenCenter. For more information on how to configure SNMP by using XenCenter, see Monitoring host and dom0 resources with SNMP.

`result` objects When configuring SNMP, all responses are returned in JSON format. If a command executes successfully, it returns the key value pair `"code": 0`. Some commands (such as the `get-config` command) return a nested JSON object called `result`. The `result` JSON object is also required for the `set-config` command which is used to update the SNMP configuration.

The `result` JSON object is made up of the following objects `common`, `agent`, and `nmss`:

`common`

Field	Allowed values	Default value
`enabled`	`no` (disable SNMP service) or `yes` (enable SNMP service)	`no`
`debug_log`	`no` (disable debug logging) or `yes` (enable debug logging)	`no`
`max_nmss`	N/A (This field is read-only and specifies the max number of supported NMSs)	1

`agent`

Field	Allowed values	Default value
`v2c`	`no` (disable SNMPv2c) or `yes` (enable SNMPv2c)	`yes`
`community`	COMMON_STRING_TYPE (see note 1)	**`public`**

Field	Allowed values	Default value
`v3`	`no` (disable v3) or `yes` (enable v3)	`no`
`user_name`	COMMON_STRING_TYPE (see note 1)	
`authentication_password`	COMMON_STRING_TYPE where length >= 8 (see note 1)	
`authentication_protocol`	`MD5` or `SHA`	
`privacy_password`	COMMON_STRING_TYPE where length >= 8 (see note 1)	
`privacy_protocol`	`DES` or `AES`	
`engine_id`	N/A (This field is read-only and is generated when the SNMP agent starts for the first time)	

`nmss`

Field	Allowed values	Default value
`uuid`	NMS UUID (You set this when you configure the NMS trap receiver and this value should be consistent across all hosts in a pool)	
`address`	NMS IPv4 address or host name (FQDN)	
`port`	1 to 65535	`162`
`v2c`	`no` (disable SNMPv2c), `yes` (enable SNMPv2c), or support either SNMPv2c or v3.	`yes`
`community`	COMMON_STRING_TYPE (see note 1)	**`public`**
`v3`	`no` (disable v3), `yes` (enable v3), or support either SNMPv2c or SNMPv3.	`no`

Field	Allowed values	Default value
`user_name`	COMMON_STRING_TYPE (see note 1)	
`authentication_password`	COMMON_STRING_TYPE where length >= 8 (see note 1)	
`authentication_protocol`	`MD5` or `SHA`	
`privacy_password`	COMMON_STRING_TYPE where length >= 8 (see note 1)	
`privacy_protocol`	`DES` or `AES`	

Notes:

1. COMMON_STRING_TYPE refers to a string that meets the following requirements:
   - Any combination of letter, number, hyphen (-), period (.), pound (#), at sign (@), equals (=), colon (:) or underscore characters (_).
   - Length between 6 and 32 inclusive.
2. Passwords are not stored in plaintext in any configuration file in XenServer. They are converted to a localized key and stored. The `get-config` command shows the password as a hidden constant comprised of asterisks (*).

Configure the SNMP service **Get the status of the SNMP service:**

```
xe host-call-plugin host-uuid=<host-uuid> plugin=snmp fn=status
<!--NeedCopy-->
```

Start, stop, or restart the SNMP service:

```
xe host-call-plugin host-uuid=<host-uuid> plugin=snmp fn=<operation>
<!--NeedCopy-->
```

where **operation** is `start`, `stop`, or `restart`.

Get the SNMP configuration details:

```
xe host-call-plugin host-uuid=<host-uuid> plugin=snmp fn=get-config
<!--NeedCopy-->
```

If successful, this command returns the key value pair `"code": 0` and the `result` JSON object containing the configuration details of the SNMP service. For example:

```
"code": 0,
  "result": {

    "common": {

      "enabled": "no",
      "debug_log": "no",
      "max_nmss": 1
     }
 ,
    "agent": {

      "v2c": "yes",
      "v3": "no",
      "community": "public",
      "user_name": "",
      "authentication_password": "",
      "authentication_protocol": "",
      "privacy_password": "",
      "privacy_protocol": "",
      "engine_id": "<engine_id>"
     }
 ,
    "nmss": []
   }

<!--NeedCopy-->
```

Copy the `result` JSON object to your preferred text editor and remove all newline (**\n**) characters from the file. Update the fields with your SNMP configuration details. Configure your NMS by referring to your NMS documentation and specifying values for the fields required for the `nmss` object. For more information, refer to the objects listed above.

To configure the SNMP service, run the `set-config` command and provide the edited `result` JSON object as a parameter value to the `args:config` parameter.

Set the SNMP configuration:

```
xe host-call-plugin host-uuid=<host-uuid> plugin=snmp fn=set-config
    args:config='<result>'
<!--NeedCopy-->
```

where **result** is the `result` JSON object returned from the `get-config` command that you copied and edited.

> **Note:**
>
> To configure SNMP for an entire pool, you must run the `set-config` command for each host in the pool.

If the configuration changes are successful, the command returns the key value pair `"code": 0`. If the configuration changes are unsuccessful, the `set-config` command returns one of the following key value pairs which indicate that an error has occurred:

- `"code":` 1: Common error string. For example, an unknown exception.
- `"code":` 2: Error string (parameter is missing).
- `"code":` 3: Returns a `message` object as a list where each element is in the format of `[ field_path, key, value, error string]`.

You can also send a test SNMP trap to your NMS to verify that the specified trap receiver information is correct.

Send a test SNMP trap:

```
xe host-call-plugin host-uuid=<host-uuid> plugin=snmp fn=send-test-trap
    args:config='{
 "nmss":[{
 "uuid":"<uuid>","address":"<address>","port":162,"v2c":"yes","v3":"no
    ","community":"public","user_name":"<user_name>","
    authentication_password":"<authentication_password>","
    authentication_protocol":"<authentication_protocol>","
    privacy_password":"<privacy_password>","privacy_protocol":"<
    privacy_protocol>" }
 ] }
 '
<!--NeedCopy-->
```

This command sends a test trap to your NMS with the `msg_name` of `TEST_TRAP` and the `msg_body` of `This is a test trap from XenServer pool "<pool name>"to verify the NMS Trap Receiver configuration`.

If you do not receive the test trap, check your SNMP configuration again. If unsuccessful, the `send-test-trap` command also returns one of the following key value pairs which indicate that an error has occurred:

- `"code":` 1: Common error string. For example, an unknown exception.
- `"code":` 2: Error string (parameter is missing).
- `"code":` 3: Returns a `message` object as a list where each element is in the format of `[ field_path, key, value, error string]`.
- `"code":` 4: Returns a `message` object as a list where each element is in the format of `[nms address, nms port, error string]`.

Monitor CPU usage

The optimum number of vCPUs per pCPU on a host depends on your use case. During operation, ensure that you monitor the performance of your XenServer environment and adjust your configuration accordingly.

Terms

In this area, there are various terms that are sometimes used interchangeably. In this article, we use the following terms and meanings:

- **CPU (physical CPU):** The physical hardware attached to a processor socket.
- **Core:** A physical processing unit, capable of one independent thread of execution, which contains all functional units required to support that execution.
- **Hyperthread:** A physical processing unit, capable of one independent thread of execution, which shares some functional units with another hyperthread (also known as its "sibling thread").
- **Logical CPU (pCPU):** A unit capable of an independent thread of execution that includes a set of registers and an instruction pointer. In a system with hyperthreads enabled, this is a hyperthread. In other cases, it's a core.
- **Host pCPUs**: The total number of logical CPUs in the host.
- **vCPU (Virtual CPU)**: A virtualized logical CPU. This is a logical unit capable of an independent thread of execution, provided to VMs. In XenServer, vCPUs can "time-share"pCPUs, using a scheduler to determine which vCPU is running on which pCPU at any given time.
- **Guest vCPUs**: The vCPUs that are presented to a guest operating system inside a VM.
- **Dom0 vCPUs**: The vCPUs that are visible to the XenServer control domain (dom0).
- **Host total vCPUs**: The sum of dom0 vCPUs and all the guest vCPUs in the host.

General behavior

The total number of vCPUs on a host is the number of vCPUs used by dom0 added to the total number of vCPUs assigned to all the VMs on the host. As you increase the number of vCPUs on a host, you can experience the following types of behavior:

- When the total number of vCPUs on the host is *less than or equal to* the number of pCPUs on the host, the host always provides as much CPU as is requested by the VMs.

- When the total number of vCPUs on the host is *greater than* the number of pCPUs on the host, the host shares the time of the host pCPUs to the VMs. This behavior does not generally affect the VMs because their vCPUs are usually idle for some of the time and, in most cases the host does not reach 100% pCPU usage.

- When the total number of vCPUs on the host is *greater than* the number of pCPUs on the host and the host is *sometimes* reaching 100% host pCPU usage, the vCPUs of the VMs don't receive as much pCPU as they request during the spikes. Instead, during these spikes the VMs slow down to receive a share of the available pCPU on the host.

- When the total number of vCPUs on the host is *greater than* the number of pCPUs on the host and the host is *often* reaching 100% host pCPU usage, the vCPUs of the VMs are continuously slowed down to receive a share of the available CPUs on the host. If the VMs have real-time requirements, this situation is not ideal and you can address it by reducing the number of vCPUs on the host.

The optimum number of vCPUs on a host can depend on the VM users' perception of the speed of their VMs, especially when the VMs have real-time requirements.

Getting information about your CPUs

To find the total number of pCPUs on your host, run the following command:

```
xe host-cpu-info --minimal
```

To find the total number of vCPUs (guest and dom0) currently on your host, run the following command:

```
xl vcpu-list | grep -v VCPU | wc -l
```

Monitoring CPU usage with RRD metrics

XenServer provides RRD metrics that describe how the vCPUs on your VMs are performing.

When host pCPU usage is 100%

When a host is reaching 100% of host pCPU usage, use these VM metrics to decide whether to move the VM to another host:

runstate_concurrency_hazard

- **runstate_concurrency_hazard > 0%** indicates that sometimes, at least one vCPU is running while at least one other vCPU wants to run but can't get pCPU time. If the vCPUs must coordinate, this behavior causes performance issues.
- **runstate_concurrency_hazard approaching 100%** is a situation to avoid.

  > **Suggested actions:**
  >
  > If there are performance issues, take one of the following actions:

> - Decrease the number of vCPUs in the VM.
> - Move the VM to another host.
> - Decrease the total number of vCPUs on the host by migrating other VMs or decreasing their number of vCPUs.

runstate_partial_contention

- **runstate_partial_contention > 0%** indicates both that at least one vCPU wants to run but can't get pCPU time, and also that at least one other vCPU is blocked (either because there's nothing to do or it's waiting for I/O to complete).
- **runstate_concurrency_hazard approaching 100%** is a situation to avoid.

  > **Suggested action:**
  >
  > Check whether the back end I/O storage servers are overloaded by looking at the back-end metrics provided by your storage vendor. If the storage servers are not overloaded and there are performance issues, take one of the following actions:
  >
  > - Decrease the number of vCPUs in the VM.
  > - Move the VM to another host.
  > - Decrease the total number of vCPUs on the host by migrating other VMs or decreasing their number of vCPUs.

runstate_full_contention

- **runstate_full_contention > 0%** indicates that sometimes the vCPUs want to run all at the same time but none can get pCPU time.
- **runstate_full_contention approaching 100%** is a situation to avoid.

  > **Suggested actions:**
  >
  > If there are performance issues, take one of the following actions:
  >
  > - Decrease the number of vCPUs in the VM.
  > - Move the VM to another host.
  > - Decrease the total number of vCPUs on the host by migrating other VMs or decreasing their number of vCPUs.

When host pCPU usage is less than 100%

If a host is not reaching 100% of host pCPU usage, use these VM metrics to decide whether a VM has the right number of vCPUs:

runstate_fullrun

- **runstate_fullrun = 0%** indicates that the vCPUs are never being used all at the same time.

  > **Suggested action:**
  >
  > Decrease the number of vCPUs in this VM.

- **0% < runstate_fullrun < 100%** indicates that the vCPUs are sometimes being used all at the same time.
- **runstate_fullrun = 100%** indicates that the vCPUs are always being used all at the same time.

  > **Suggested action:**
  >
  > You can increase the number of vCPUs in this VM, until runstate_fullrun < 100%. Do not increase the number of vCPUs further, otherwise it can increase the probability of concurrency hazard if the host reaches 100% of pCPU usage.

runstate_partial_run

- **runstate_partial_run = 0%** indicates that either all vCPUs are always being used (fullrun=100%) or no vCPUs are being used (idle=100%).
- **0% < runstate_partial_run < 100%** indicates that, sometimes, at least one vCPU is blocked, either because they have nothing to do, or because they are waiting for I/O to complete.
- **runstate_partial_run=100%** indicates that there is always at least one vCPU that is blocked.

  > **Suggested action**:
  >
  > Check whether the back-end I/O storage servers are overloaded. If they are not, the VM probably has too many vCPUs and you can decrease the number of vCPUs in this VM. Having too many vCPUs in a VM can increase the risk of the VM going into the concurrency hazard state when the host CPU usage reaches 100%.

Manage virtual machines

May 7, 2024

This section provides an overview of how to create Virtual Machines (VMs) using templates. It also explains other preparation methods, including cloning templates and importing previously exported VMs.

What is a virtual machine?

A Virtual Machine (VM) is a software computer that, like a physical computer, runs an operating system and applications. The VM comprises a set of specification and configuration files backed by the physical resources of a host. Every VM has virtual devices that provide the same functions as physical hardware. VMs can give the benefits of being more portable, more manageable, and more secure. In addition, you can tailor the boot behavior of each VM to your specific requirements. For more information, see VM Boot Behavior.

XenServer supports guests with any combination of IPv4 or IPv6 configured addresses.

In XenServer VMs can operate in full virtualized mode. Specific processor features are used to 'trap' privileged instructions that the virtual machine carries out. This capability enables you to use an unmodified operating system. For network and storage access, emulated devices are presented to the virtual machine. Alternatively, PV drivers can be used for performance and reliability reasons.

Create VMs

Use VM templates

VMs are prepared from templates. A template is a *gold image* that contains all the various configuration settings to create an instance of a specific VM. XenServer ships with a base set of templates, which are *raw* VMs, on which you can install an operating system. Different operating systems require different settings to run at their best. XenServer templates are tuned to maximize operating system performance.

There are two basic methods by which you can create VMs from templates:

- Using a complete pre-configured template.
- Installing an operating system from a CD, ISO image or network repository onto the appropriate provided template.

Windows VMs describes how to install Windows operating systems onto VMs.

Linux VMs describes how to install Linux operating systems onto VMs.

> **Note:**
>
> Templates created by older versions of XenServer can be used in newer versions of XenServer. However, templates created in newer versions of XenServer are not compatible with older versions of XenServer. If you created a VM template by using Citrix Hypervisor 8.2, to use it with an earlier version, export the VDIs separately and create the VM again.

Other methods of VM creation

In addition to creating VMs from the provided templates, you can use the following methods to create VMs.

Clone an existing VM You can make a copy of an existing VM by *cloning* from a template. Templates are ordinary VMs which are intended to be used as original copies to create instances of VMs from. A VM can be customized and converted into a template. Ensure that you follow the appropriate preparation procedure for the VM. For more information, see Preparing for Cloning a Windows VM Using Sysprep and Preparing to Clone a Linux VM.

> **Note:**
>
> Templates cannot be used as normal VMs.

XenServer has two mechanisms for cloning VMs:

- A full copy
- Copy-on-Write

  The faster Copy-on-Write mode only writes *modified* blocks to disk. Copy-on-Write is designed to save disk space and allow fast clones, but slightly slows down normal disk performance. A template can be fast-cloned multiple times without slowdown.

  > **Note:**
  >
  > If you clone a template into a VM and then convert the clone into a template, disk performance can decrease. The amount of decrease has a linear relationship to the number of times this process has happened. In this event, the `vm-copy` CLI command can be used to perform a full copy of the disks and restore expected levels of disk performance.

Notes for resource pools If you create a template from VM virtual disks on a shared SR, the template cloning operation is forwarded to any host in the pool that can access the shared SRs. However, if you create the template from a VM virtual disk that only has a local SR, the template clone operation is only able to run on the host that can access that SR.

Import an exported VM You can create a VM by *importing* an existing exported VM. Like cloning, exporting and importing a VM is fast way to create more VMs of a certain configuration. Using this method enables you to increase the speed of your deployment. You might, for example, have a special-purpose host configuration that you use many times. After you set up a VM as required, export it and import it later to create another copy of your specially configured VM. You can also use export and import to move a VM to the XenServer host that is in another resource pool.

For details and procedures on importing and exporting VMs, see Importing and Exporting VMs.

XenServer VM Tools

XenServer VM Tools provide high performance I/O services without the overhead of traditional device emulation.

XenServer VM Tools for Windows

XenServer VM Tools for Windows consist of I/O drivers (also known as paravirtualized drivers or PV drivers) and the Management Agent.

The I/O drivers contain storage and network drivers, and low-level management interfaces. These drivers replace the emulated devices and provide high-speed transport between Windows and the XenServer product family software. While installing a Windows operating system, XenServer uses traditional device emulation to present a standard IDE controller and a standard network card to the VM. This emulation allows the Windows installation to use built-in drivers, but with reduced performance due to the overhead inherent in emulating the controller drivers.

The Management Agent, also known as the Guest Agent, is responsible for high-level virtual machine management features and provides a full set of functions to XenCenter.

Install XenServer VM Tools for Windows on each Windows VM for that VM to have a fully supported configuration, and to be able to use the xe CLI or XenCenter. A VM functions without the XenServer VM Tools for Windows, but performance is hampered when the I/O drivers (PV drivers) are not installed. You must install XenServer VM Tools for Windows on Windows VMs to be able to perform the following operations:

- Cleanly shut down, reboot, or suspend a VM
- View VM performance data in XenCenter
- Migrate a running VM (using live migration or storage live migration)
- Create snapshots with memory (checkpoints) or revert to snapshots

For more information, see Install XenServer VM Tools for Windows.

XenServer VM Tools for Linux

XenServer VM Tools for Linux contain a guest agent that provides extra information about the VM to the host.

You must install the XenServer VM Tools for Linux on Linux VMs to be able to perform the following operations:

- View VM performance data in XenCenter
- Adjust the number of vCPUs on a running Linux VM
- Enable dynamic memory control

> **Note:**
>
> You cannot use the Dynamic Memory Control (DMC) feature on Red Hat Enterprise Linux 8, Red Hat Enterprise Linux 9, Rocky Linux 8, Rocky Linux 9, or CentOS Stream 9 VMs as these operating systems do not support memory ballooning with the Xen hypervisor.

For more information, see Install XenServer VM Tools for Linux.

Find out the virtualization state of a VM

XenCenter reports the virtualization state of a VM on the VM's **General** tab. You can find out whether or not XenServer VM Tools are installed. This tab also displays whether the VM can install and receive updates from Windows Update. The following section lists the messages displayed in XenCenter:

I/O optimized (not optimized): This field displays whether or not the I/O drivers are installed on the VM.

Management Agent installed (not installed): This field displays whether or not the Management Agent is installed on the VM.

Able to (Not able to) receive updates from Windows Update: specifies whether the VM can receive I/O drivers from Windows Update.

> **Note:**
>
> Windows Server Core 2016 does not support using Windows Update to install or update the I/O drivers. Instead use the XenServer VM Tools for Windows installer provided on the XenServer Downloads page.

Install I/O drivers and Management Agent: this message is displayed when the VM does not have the I/O drivers or the Management Agent installed.

Guest UEFI boot and Secure Boot

XenServer enables the following guest operating systems to boot in UEFI mode:

- Windows 10
- Windows 11
- Windows Server 2016

- Windows Server 2019
- Windows Server 2022
- Red Hat Enterprise Linux 8
- Red Hat Enterprise Linux 9 (preview)
- Ubuntu 20.04
- Ubuntu 22.04
- Rocky Linux 8
- Rocky Linux 9 (preview)
- SUSE Linux Enterprise 15
- Debian Bookworm 12 (preview)
- Oracle Linux 8

UEFI boot provides a richer interface for the guest operating systems to interact with the hardware, which can significantly reduce VM boot times. If XenServer supports UEFI boot for your guest operating system, we recommend that you choose this boot mode instead of BIOS.

For these operating systems, XenServer also supports Secure Boot. Secure Boot prevents unsigned, incorrectly signed or modified binaries from being run during boot. On a UEFI-enabled VM that enforces Secure Boot, all drivers must be signed. This requirement might limit the range of uses for the VM, but provides the security of blocking unsigned/modified drivers. If you use an unsigned driver, secure boot fails and an alert is shown in XenCenter. Secure Boot also reduces the risk that malware in the guest can manipulate the boot files or run during the boot process.

You must specify the boot mode when creating a VM. It is not possible to change the boot mode of a VM between BIOS and UEFI (or UEFI Secure Boot) after booting the VM for the first time. However, you can change the boot mode between UEFI and UEFI Secure Boot after the VM is used to troubleshoot potential Secure Boot issues. For more information, see Troubleshooting.

Consider the following when enabling UEFI boot on VMs:

- Ensure that a UEFI-enabled Windows VM has at least two vCPUs. UEFI-enabled Linux VMs do not have this restriction.
- You can import or export a UEFI-enabled VM created on XenServer as an OVA, OVF, or an XVA file. Importing a UEFI-enabled VM from OVA or OVF packages created on other hypervisors is not supported.
- To use PVS-Accelerator with UEFI-enabled VMs, ensure that you are using Citrix Provisioning 1906 or later.
- For Windows VMs, use the UEFI settings menu to change the screen resolution of the XenCenter console. For detailed instructions, see Troubleshooting.

> **Note**
>
> UEFI-enabled VMs use NVME and E1000 for emulated devices. The emulation information does

> not display these values until after you install XenServer VM Tools for Windows on the VM. UEFI-enabled VMs also show as only having 2 NICs until after you install XenServer VM Tools for Windows.

Enabling UEFI boot or UEFI Secure Boot

You can use XenCenter or the xe CLI to enable UEFI boot or UEFI Secure Boot for your VM.

For information about creating a UEFI-enabled VM in XenCenter, see Create a Windows VM by using XenCenter or Create a Linux VM by using XenCenter.

Using the xe CLI to enable UEFI boot or UEFI Secure Boot When you create a VM, run the following command before booting the VM for the first time:

```
    xe vm-param-set uuid=<UUID> HVM-boot-params:firmware=<MODE>
    xe vm-param-set uuid=<UUID> platform:device-model=qemu-upstream-
       uefi
    xe vm-param-set uuid=<UUID> platform:secureboot=<OPTION>
<!--NeedCopy-->
```

Where, `UUID` is the VM's UUID, `MODE` is either `BIOS` or `uefi`, and `OPTION` is either 'true'or 'false'. If you do not specify the mode, it defaults to `uefi` if that option is supported for your VM operating system. Otherwise, the mode defaults to `BIOS`. If you do not specify the `secureboot` option, it defaults to 'auto'. For UEFI-enabled VMs, the 'auto'behavior is to enable Secure Boot for the VM.

To create a UEFI-enabled VM from a template supplied with XenServer, run the following command:

```
    UUID=$(xe vm-clone name-label='Windows 10 (64-bit)' new-name-label=
       'Windows 10 (64-bit)(UEFI)')
    xe template-param-set uuid=<UUID> HVM-boot-params:firmware=<MODE>
       platform:secureboot=<OPTION>
<!--NeedCopy-->
```

Do not run this command for templates that have something installed on them or templates that you created from a snapshot. The boot mode of these snapshots cannot be changed and, if you attempt to change the boot mode, the VM fails to boot.

When you boot the UEFI-enabled VM the first time you are prompted on the VM console to press any key to start the installation. If you do not start the operating system installation, the VM console switches to the UEFI shell.

To restart the installation process, in the UEFI console, type the following commands.

```
EFI:
EFI\BOOT\BOOTX64
```

When the installation process restarts, watch the VM console for the installation prompt. When the prompt appears, press any key.

Disabling Secure Boot

You might want to disable Secure Boot on occasion. For example, some types of debugging cannot be enabled on a VM that in Secure Boot user mode. To disable Secure Boot, change the VM into Secure Boot setup mode. On your XenServer host, run the following command:

```
varstore-sb-state <VM_UUID> setup
```

Keys

For Windows VMs:

UEFI-enabled Windows VMs are provisioned with a PK from an ephemeral private key, the Microsoft KEK, the Microsoft Windows Production PCA, and Microsoft third party keys. The VMs are also provided with an up-to-date revocation list from the UEFI forum. This configuration enables Windows VMs to boot with Secure Boot turned on and to receive automatic updates to the keys and revocation list from Microsoft.

For Linux VMs:

To install third-party drivers in a Linux VM that has Secure Boot enabled, you must create a signing key, add it to the VM as a machine owner key (MOK), and use that key to sign the driver. For more information, see Install third-party drivers on your Secure Boot Linux VM.

Troubleshooting your UEFI and UEFI Secure Boot VMs

For information about troubleshooting your UEFI or UEFI Secure Boot VMs, see Troubleshoot UEFI and Secure Boot problems.

Supported guests and allocating resources

For a list of supported guest operating systems, see Supported Guests, Virtual Memory, and Disk Size Limits

This section describes the differences in virtual device support for the members of the XenServer product family.

XenServer product family virtual device support

The current version of the XenServer product family has some general limitations on virtual devices for VMs. Specific guest operating systems may have lower limits for certain features. The individual guest installation section notes the limitations. For detailed information on configuration limits, see Configuration Limits.

Factors such as hardware and environment can affect the limitations. For information about supported hardware, see the XenServer Hardware Compatibility List.

VM block devices XenServer emulates an IDE bus in the form of an `hd*` device. When using Windows, installing the XenServer VM Tools installs a special I/O driver that works in a similar way to Linux, except in a fully virtualized environment.

CPU features

A pool's CPU feature set can change while a VM is running, for example, when a new host is added to an existing pool or when the VM is migrated to a host in another pool. When a pool's CPU feature set changes, the VM continues to use the feature set which was applied when it was started. To update the VM to use the pool's new feature set, you must restart the VM.

Windows VMs

April 29, 2024

Installing Windows VMs on the XenServer host requires hardware virtualization support (Intel VT or AMD-V).

> **Note:**
>
> Nested virtualization is not supported for Windows VMs hosted on XenServer.

Basic procedure for creating a Windows VM

The process of installing a Windows on to a VM consists of the following steps:

1. Selecting the appropriate Windows template
2. Choosing the appropriate boot mode
3. Installing the Windows operating system

4. Installing the XenServer VM Tools for Windows (*I/O drivers* and the *Management Agent*)

> **Warning:**
>
> Windows VMs are supported only when the VMs have the XenServer VM Tools for Windows installed.

Windows VM templates

Windows operating systems are installed onto VMs by cloning an appropriate template using either XenCenter or the xe CLI, and then installing the operating system. The templates for individual guests have predefined platform flags set which define the configuration of the virtual hardware. For example, all Windows VMs are installed with the ACPI Hardware Abstraction Layer (HAL) mode enabled. If you later change one of these VMs to have multiple virtual CPUs, Windows automatically switches the HAL to multi-processor mode.

The available Windows templates are listed in the following table:

Template Name	Supported boot modes	Description
Windows 10 (64-bit)	BIOS, UEFI, UEFI Secure Boot	Used to install Windows 10 (64-bit)
Windows 11 (64-bit)	UEFI, UEFI Secure Boot	Used to install Windows 11 (64-bit)
Windows Server 2016 (64-bit)	BIOS, UEFI, UEFI Secure Boot	Used to install Windows Server 2016 or Windows Server Core 2016 (64-bit)
Windows Server 2019 (64-bit)	BIOS, UEFI, UEFI Secure Boot	Used to install Windows Server 2019 or Windows Server Core 2019 (64-bit)
Windows Server 2022 (64-bit)	BIOS, UEFI, UEFI Secure Boot	Used to install Windows Server 2022 or Windows Server Core 2022 (64-bit)

XenServer supports all SKUs (editions) for the listed versions of Windows.

Attach an ISO image library

The Windows operating system can be installed either from an install CD in a physical CD-ROM drive on the XenServer host, or from an ISO image. See Create ISO images for information on how to make an ISO image from a Windows install CD and make it available for use.

Create a VM by using XenCenter

To create a Windows VM:

1. On the XenCenter toolbar, click the **New VM** button to open the New VM wizard.

   The New VM wizard allows you to configure the new VM, adjusting various parameters for CPU, storage, and networking resources.

2. Select a VM template and click **Next**.

   Each template contains the setup information that is required to create a VM with a specific guest operating system (OS), and with optimum storage. This list reflects the templates that XenServer currently supports.

   > **Note:**
   >
   > If the OS that you are installing on your VM is compatible only with the original hardware, check the **Copy host BIOS strings to VM** box. For example, you might use this option for an OS installation CD that was packaged with a specific computer.
   >
   > After you first start a VM, you cannot change its BIOS strings. Ensure that the BIOS strings are correct before starting the VM for the first time.

   To copy BIOS strings using the CLI, see Install VMs from Reseller Option Kit (BIOS-locked) Media.

   Advanced users can set user-defined BIOS strings. For more information, see User-defined BIOS strings.

3. Enter a name and an optional description for the new VM.

4. Choose the source of the OS media to install on the new VM.

   Installing from a CD/DVD is the simplest option for getting started.

   a) Choose the default installation source option (DVD drive)
   b) Insert the disk into the DVD drive of the XenServer host

   XenServer also allows you to pull OS installation media from a range of sources, including a pre-existing ISO library. An ISO image is a file that contains all the information that an optical disc (CD, DVD, and so on) would contain. In this case, an ISO image would contain the same OS data as a Windows installation CD.

   To attach a pre-existing ISO library, click **New ISO library** and indicate the location and type of the ISO library. You can then choose the specific operating system ISO media from the list.

5. In the **Installation Media** tab, you can choose a boot mode for the VM. By default, XenCenter selects the most secure boot mode available for the VM operating system version.

> **Notes:**
>
> - The **UEFI Boot** and **UEFI Secure Boot** options appear grayed out if the VM template you have chosen does not support UEFI boot.
> - You cannot change the boot mode after you boot the VM for the first time.
>
> For more information, see Guest UEFI boot and Secure Boot.

6. If required, change the option **Create and attach a new vTPM**.

    - For VM operating systems that require a vTPM, the option is selected and cannot be unselected.
    - For VM operating systems that do not support a vTPM, the option is grayed out and cannot be selected.
    - For VM operating systems that support vTPM, but do not require it, choose whether to attach a vTPM to the VM.

    For more information, see vTPM.

7. Select a home server for the VM.

    A home server is the host which provides the resources for a VM in a pool. When you nominate a home server for a VM, XenServer attempts to start the VM on that host. If this action is not possible, an alternate host within the same pool is selected automatically. To choose a home server, click **Place the VM on this server** and select a host from the list.

    > **Notes:**
    >
    > - In WLB-enabled pools, the nominated home server isn't used for starting, restarting, resuming, or migrating the VM. Instead, Workload Balancing nominates the best host for the VM by analyzing XenServer resource pool metrics and by recommending optimizations.
    > - If a VM has one or more virtual GPUs assigned to it, the home server nomination doesn't take effect. Instead, the host nomination is based on the virtual GPU placement policy set by the user.
    > - During rolling pool upgrade, the home server is not considered when migrating the VM. Instead, the VM is migrated back to the host it was on before the upgrade.

    If you do not want to nominate a home server, click **Don't assign this VM a home server**. The VM is started on any host with the necessary resources.

    Click **Next** to continue.

8. Allocate processor and memory resources for the VM. For a Windows 10 VM (64-bit), the defaults are 2 virtual CPUs and 4 GB of RAM. You can also choose to modify the defaults. Click **Next** to continue.

9. Assign a virtual GPU. The New VM wizard prompts you to assign a dedicated GPU or one or more virtual GPUs to the VM. This option enables the VM to use the processing power of the GPU. With this feature, you have better support for high-end 3D professional graphics applications such as CAD/CAM, GIS, and Medical Imaging applications.

10. Allocate and configure storage for the new VM.

    Click **Next** to select the default allocation (32 GB) and configuration, or you might want to do the following extra configuration:

    - Change the name, description, or size of your virtual disk by clicking **Edit**.
    - Add a new virtual disk by selecting **Add**.

11. Configure networking on the new VM.

    Click **Next** to select the default NIC and configurations, including an automatically created unique MAC address for each NIC. Alternatively, you might want to do the following extra configuration:

    - Change the physical network, MAC address, or Quality of Service (QoS) priority of the virtual disk by clicking **Edit**.
    - Add a new virtual NIC by selecting **Add**.

12. (Optional) If this VM is to be used as a template with Citrix Provisioning or with the `reset-on-boot` flag set, ensure **Start the new VM automatically** is not selected. This enables you to do some required configuration before installing Windows.

13. Review settings, and then click **Create Now** to create the VM and return to the **Search** tab.

    An icon for your new VM appears under the host in the **Resources** pane.

14. (Optional) If this VM is intended to be used as a template with Citrix Provisioning or has the `reset-on-boot` flag set, configure the VM before installing Windows.

    In the host console, type the following command:

    ```
    xe vm-param-set uuid=<uuid> has-vendor-device=false
    ```

    The flag `has-vendor-device=false` ensures that Windows Update does not attempt to install or update the I/O drivers included in the XenServer VM Tools. For more information, see Settings for Citrix Provisioning targets or reset-on-boot machines.

15. On the **Resources** pane, select the VM, and then click the **Console** tab to see the VM console.

16. (Optional) If you want to be able to clone the VM, we recommend that you do not run the Windows first-time setup, known as the Out-Of-Box-Experience (OOBE). Instead, when the OOBE starts on the page that asks for region information, press **Ctrl + Shift + F3** to enter audit mode.

You can then use Sysprep to generalize the VM. For more information see Prepare to clone a Windows VM by using Sysprep.

If you do not intend to clone the VM, continue to the following steps in this procedure.

17. Follow the OS installation screen and make your selections.
18. After the OS installation completes and the VM reboots, Install XenServer VM Tools for Windows.

Create a Windows VM by using the CLI

To create a Windows VM from an ISO repository by using the xe CLI:

> **Note:**
>
> For Windows 10 and Windows 11 VMs, the requirement for a vTPM is specified by the template. You are not required to add anything to the xe CLI commands to set up the vTPM.

1. Create a VM from a template:

   ```
   xe vm-install new-name-label=<vm_name> template=<template_name>
   <!--NeedCopy-->
   ```

   This command returns the UUID of the new VM.

2. (Optional) Change the boot mode of the VM.

   ```
   xe vm-param-set uuid=<uuid> HVM-boot-params:firmware=<mode>
   xe vm-param-set uuid=<uuid> platform:secureboot=<option>
   <!--NeedCopy-->
   ```

   The value of `mode` can be either `BIOS` or `uefi` and defaults to `uefi` if that option is supported for your VM operating system. Otherwise, the mode defaults to `BIOS`. The value of `option` can be set to either **`true`** or **`false`**. If you do not specify the Secure Boot option, it defaults to `auto`.

   For more information, see Guest UEFI boot and Secure Boot.

3. (Optional) If this VM is intended to be used as a template with Citrix Provisioning or has the `reset-on-boot` flag set, configure the VM before installing Windows.

   ```
   xe vm-param-set uuid=<uuid> has-vendor-device=false
   <!--NeedCopy-->
   ```

   The flag `has-vendor-device=`**`false`** ensures that Windows Update does not attempt to install or update the I/O drivers included in the XenServer VM Tools. For more information, see Settings for Citrix Provisioning targets or reset-on-boot machines.

4. Create an ISO storage repository:

```
xe-mount-iso-sr <path_to_iso_sr>
<!--NeedCopy-->
```

5. List all of the available ISOs:

```
xe cd-list
<!--NeedCopy-->
```

6. Insert the specified ISO into the virtual CD drive of the specified VM:

```
xe vm-cd-add vm=<vm_name> cd-name=<iso_name> device=3
<!--NeedCopy-->
```

7. Start the VM and install the operating system:

```
xe vm-start vm=<vm_name>
<!--NeedCopy-->
```

   At this point, the VM console is visible in XenCenter.

8. On the XenCenter **Resources** pane, select the VM, and then click the **Console** tab to see the VM console.

9. (Optional) If you want to be able to clone the VM, we recommend that you do not run the Windows first-time setup, known as the Out-Of-Box-Experience (OOBE). Instead, when the OOBE starts on the page that asks for region information, press **Ctrl + Shift + F3** to enter audit mode.

   You can then use Sysprep to generalize the VM. For more information see Prepare to clone a Windows VM by using Sysprep.

   If you do not intend to clone the VM, continue to the following steps in this procedure.

10. Follow the OS installation screen and make your selections.

11. After the OS installation completes and the VM reboots, install the XenServer VM Tools for Windows.

For more information on using the CLI, see Command line interface.

Install XenServer VM Tools for Windows

XenServer VM Tools for Windows provide high performance I/O services without the overhead of traditional device emulation. For more information about the XenServer VM Tools for Windows and advanced usage, see XenServer VM Tools for Windows.

> **Note:**

> To install XenServer VM Tools for Windows on a Windows VM, the VM must be running the Microsoft .NET Framework Version 4.0 or later.

Before you install the XenServer VM Tools for Windows, ensure that your VM is configured to receive the I/O drivers from Windows Update. Windows Update is the recommended way to receive updates to the I/O drivers. However, if Windows Update is not an available option for your VM, you can also receive updates to the I/O drivers through other means. For more information, see XenServer VM Tools for Windows.

To install XenServer VM Tools for Windows:

1. We recommend that you snapshot your VM before installing or updating the XenServer VM Tools.
2. Download the XenServer VM Tools for Windows file from the XenServer Downloads page.
3. Verify your download against the provided SHA256 value.
4. Copy the file to your Windows VM or to a shared drive that the Windows VM can access.
5. Run the `managementagentxXX.msi` file to begin XenServer VM Tools installation.

   ```
   Msiexec.exe /package managementagentxXX.msi
   ```

6. Follow the prompts in the installer.

   a) Follow the instructions on the wizard to accept the license agreement and choose a destination folder.

   b) The wizard displays the recommended settings on the **Installation and Updates Settings** page. For information about customizing these settings, see XenServer VM Tools for Windows.

   c) Click **Next** and then **Install** to begin the XenServer VM Tools for Windows installation process.

7. Restart the VM when prompted to complete the installation process.

vTPM

XenServer enables you to create a virtual Trusted Platform Module (vTPM) and attach it to your Windows 10 or Windows 11 VM.

Windows 11 VMs require the presence of a linked vTPM. This vTPM is created automatically when the Windows 11 VM is created from the provided template. For Windows 10 VMs the vTPM is optional.

A VM has a one-to-one relationship with its linked vTPM. A VM can only have one vTPM and a vTPM can only be associated with a single VM. Users with the VM Admin and above roles can create and destroy vTPM instances.

Applications running on the VM can access the vTPM through the TPM 2.0 compliant API. TPM 1.2 is not supported. Users with the VM Operator and above roles can access the vTPM through the VM.

To check if your VM has a linked vTPM, in XenCenter go to the **General** tab and look in the **Device Security** section.

Constraints

The following constraints currently apply to VMs created with an attached vTPM:

- While you can export your VMs to OVF/OVA format, any data in your vTPM is lost as part of this process. This lost data can cause the VM to show unexpected behaviors or can prevent it from starting. If you are using any vTPM features in the VM, do not export your VMs using this format.
- BitLocker is not currently supported for VMs with a vTPM attached.
- HA is not currently supported for VMs with a vTPM attached.

Known issues

- If you have a lot of VMs with a vTPM, you can experience the following behaviors:
  - The XAPI database becomes large and consumes a lot of memory.
  - VM writes to the vTPM can cause a bottleneck in the toolstack.
- vTPM operations performed by the user or by Windows in the background might fail in the following situations:
  - If the toolstack or the XenServer host crashes before the operation is synchronized to the disk. Errors when writing to disk are ignored.

  In the case of this type of failure the vTPM returns an error to the operating system. Windows logs these errors to the System Event log.

Attach a vTPM to a Windows VM

For new Windows 11 VMs and Windows 10 VMs, the vTPM can be added during VM creation. For more information, see the documentation for your preferred VM creation method.

If you have an existing UEFI or UEFI Secure Boot Windows 10 VM that you want to add a vTPM to, you can do it by using XenCenter or by using the xe CLI. If you want to upgrade the operating system of the VM to one that requires a vTPM, you must attach the vTPM to the VM *before* you upgrade your VM operating system.

By using XenCenter

1. Shut down the Windows 10 VM.
2. Add a vTPM to the VM.
    a) Right-click on the VM and select **Manage vTPMs**. Or, on the main menu bar, go to **VM > Manage vTPMs**. The **TPM Manager** dialog opens.
    b) In the **TPM Manager** dialog, add a vTPM.
3. To verify that the VM has a linked vTPM, select the VM and go to its **General** tab and look in the **Device Security** section.
4. Start the Windows 10 VM.

By using the xe CLI

1. Shut down the VM:

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

2. Create a vTPM and attach it to the VM:

    ```
    xe vtpm-create vm-uuid=<vm_uuid>
    <!--NeedCopy-->
    ```

3. Start the VM:

    ```
    xe vm-start uuid=<vm_uuid>
    <!--NeedCopy-->
    ```

Upgrade the Windows operating system in your VM

Upgrades to VMs are typically required when moving to a newer version of XenServer.

Before you upgrade your Windows VM

1. If you are updating your operating system to a version of Windows that requires a vTPM (such as Windows 11), you must attach a vTPM to your VM before upgrading its operating system. For more information, see Attach a vTPM to a Windows VM.
2. Upgrade XenServer VM Tools for Windows to the latest version on the VM. For more information, see XenServer VM Tools for Windows.

    We recommend that you do not remove the XenServer VM Tools from your Windows VM before automatically updating the version of Windows on the VM.

Upgrade the Windows operating system

You can upgrade your Windows VMs in one of the following ways:

- Use Windows Update to upgrade the version of the Windows operating system on your Windows VMs. If you use Windows Update to update your XenServer VM Tools, we recommend you use this method.
- Use the Windows installation ISO for the newer versions. Windows installation disks typically provide an upgrade option if you boot them on a server which has an earlier version of Windows already installed.

In your Windows VM console, follow the upgrade prompts provided by Windows.

Prepare to clone a Windows VM by using Sysprep

The only supported way to clone a Windows VM is by using the Windows utility `sysprep` to prepare the VM.

The `sysprep` utility changes the local computer SID to make it unique to each computer. The `sysprep` binaries are in the `C:\Windows\System32\Sysprep` folder.

For more information about using Sysprep, see Sysprep (Generalize) a Windows installation.

To run sysprep on a Windows VM:

> **Note:**
>
> On Windows 10 and 11, the Windows first-time setup or Out-Of-Box-Experience (OOBE), installs applications (such as AppX) that can interfere with the `sysprep` process. Because of this behavior, when creating a clonable VM, we recommend skipping the first-time setup and starting Windows in audit mode instead.

1. Create a Windows VM.
2. Install Windows.
3. (Optional) When the Out-Of-Box-Experience (OOBE) starts on the page that asks for region information, press **Ctrl + Shift + F3**. Windows starts in audit mode. For more information, see Boot Windows to Audit mode or OOBE.

   While not required, we recommend that you exit OOBE to avoid creating an unneeded user account on the image and to avoid issues with third-party application compatibility. If you continue with OOBE, some applications or Windows Updates that are installed during OOBE might prevent Sysprep from operating correctly.
4. Install the latest version of XenServer VM Tools for Windows.

5. Install any applications and perform any other configuration required.
6. Run `sysprep` to generalize the VM. This utility shuts down the VM when it completes.

> **Note:**
>
> Do not restart the original, generalized VM (the "source"VM) again after the `sysprep` stage. Immediately convert it to a template afterwards to prevent restarts. If the source VM is restarted, `sysprep` must be run on it again before it can be safely used to make more clones.

To clone a generalized Windows VM:

1. Using XenCenter convert the VM into a template.
2. Clone the newly created template into new VMs as required.
3. When the cloned VM starts, it completes the following actions before being available for use:
   - It gets a new SID and name
   - It runs a setup to prompt for configuration values as necessary
   - Finally, it restarts

Windows VM release notes

There are many versions and variations of Windows with different levels of support for the features provided by XenServer. This section lists notes and errata for the known differences.

General Windows issues

- When installing Windows VMs, start off with no more than three virtual disks. After the VM and XenServer VM Tools for Windows have been installed, you can add extra virtual disks. Ensure that the boot device is always one of the initial disks so that the VM can successfully boot without the XenServer VM Tools for Windows.
- When the boot mode for a Windows VM is BIOS boot, Windows formats the primary disk with a Master Boot Record (MBR). MBR limits the maximum addressable storage space of a disk to 2 TiB. To use a disk that is larger than 2 TiB with a Windows VM, do one of the following things:
  - If UEFI boot is supported for the version of Windows, ensure that you use UEFI as the boot mode for the Windows VM.
  - Create the large disk as the secondary disk for the VM and select GUID Partition Table (GPT) format.

- Multiple vCPUs are exposed as CPU sockets to Windows guests, and are subject to the licensing limitations present in the VM. The number of CPUs present in the guest can be confirmed by checking the Device Manager. The number of CPUs actually being used by Windows can be seen in the Task Manager.
- The disk enumeration order in a Windows guest might differ from the order in which they were initially added. This behavior is because of interaction between the I/O drivers and the Plug-and-Play subsystem in Windows. For example, the first disk might show up as `Disk 1`, the next disk hot plugged as `Disk 0`, a later disk as `Disk 2`, and then upwards in the expected fashion.
- A bug in the VLC player DirectX back-end replaces yellow with blue during video playback when the Windows display properties are set to 24-bit color. VLC using OpenGL as a back-end works correctly, and any other DirectX-based or OpenGL-based video player works too. It is not a problem if the guest is set to use 16-bit color rather than 24.
- The PV Ethernet Adapter reports a speed of 100 Gbps in Windows VMs. This speed is an artificial hardcoded value and is not relevant in a virtual environment because the virtual NIC is connected to a virtual switch. The Windows VM uses the full speed that is available, but the network might not be capable of the full 100 Gbps.
- If you attempt to make an insecure RDP connection to a Windows VM, this action might fail with the following error message: "This could be due to CredSSP encryption oracle remediation." This error occurs when the Credential Security Support Provider protocol (CredSSP) update is applied to only one of the client and server in the RDP connection. For more information, see https://support.microsoft.com/en-gb/help/4295591/credssp-encryption-oracle-remediation-error-when-to-rdp-to-azure-vm.

XenServer VM Tools for Windows

February 26, 2024

XenServer VM Tools (formerly Citrix VM Tools or XenServer PV Tools) for Windows provide high performance I/O services without the overhead of traditional device emulation. XenServer VM Tools for Windows consist of I/O drivers (also known as paravirtualized drivers or PV drivers) and the Management Agent.

XenServer VM Tools for Windows must be installed on each Windows VM for the VM to have a fully supported configuration. A VM functions without them, but performance is hampered.

The version of the XenServer VM Tools for Windows is updated independently of the version of XenServer. Ensure that your XenServer VM Tools for Windows are regularly updated to the latest version, both in your VMs and in any templates that you use to create your VMs. For more information about the latest version of the tools, see What's new.

Install XenServer VM Tools

> **Note:**
>
> To install XenServer VM Tools for Windows on a Windows VM, the VM must be running the Microsoft .NET Framework Version 4.0 or later.

The XenServer VM Tools for Windows are installed by default in the `C:\Program Files\XenServer\XenTools` directory on the VM.

To install XenServer VM Tools for Windows:

1. We recommend that you snapshot your VM before installing or updating the XenServer VM Tools.
2. Download the XenServer VM Tools for Windows file from the XenServer Downloads page.
3. Verify your download against the provided SHA256 value.
4. Copy the file to your Windows VM or to a shared drive that the Windows VM can access.
5. Run the `managementagentx64.msi` file to begin XenServer VM Tools installation.

   ```
   Msiexec.exe /package managementagentx64.msi
   ```

6. Follow the prompts in the installer.

   - Follow the instructions on the wizard to accept the license agreement and choose a destination folder.
   - Customize the settings on the **Installation and Updates Settings** page.

     By default, the wizard displays the following recommended settings:

     - Install I/O Drivers Now
       * If the VM has `has-vendor-device=true` set, this option is unselected because the I/O drivers have been already installed by Windows Update.
       * IF the VM has `has-vendor-device=false` set, this option is selected.
     - Allow automatic Management Agent updates
     - Disallow automatic I/O driver updates by the Management Agent
     - Send anonymous usage information to Cloud Software Group, Inc.

     For some use cases, different update settings are recommended. For more information, see Update XenServer VM Tools.

     To configure the update settings, you can make the following changes:

    - If you do not want to allow the automatic updating of the Management Agent, select **Disallow automatic Management Agent updates** from the list.
    - If you want to allow the Management Agent to update the I/O drivers automatically, select **Allow automatic I/O driver updates by the Management Agent**. However, we recommend that you use Windows Update to update the I/O drivers, not the Management Agent. If you have chosen to receive I/O driver updates through the Windows Update mechanism, do not allow the Management Agent to update the I/O drivers automatically.
    - If you do not want to share anonymous usage information with us, clear the **Send anonymous usage information to Cloud Software Group, Inc.** checkbox. The information transmitted to Cloud Software Group contains the first four characters of the UUID of the VM requesting the update. No other information relating to the VM is collected or transmitted.
  - Click **Next** and then **Install** to begin the XenServer VM Tools for Windows installation process.

7. Restart the VM when prompted to complete the installation process.

   Customers who install the XenServer VM Tools for Windows or the Management Agent through RDP might not see the restart prompt as it only appears on the Windows console session. To ensure that you restart your VM (if necessary) and to get your VM to an optimized state, specify the force restart option in RDP. The force restart option restarts the VM only if it is required to get the VM to an optimized state.

> **Warning:**
>
> Installing or upgrading the XenServer VM Tools for Windows can cause the friendly name and identifier of some network adapters to change. Any software which is configured to use a particular adapter might have to be reconfigured following XenServer VM Tools for Windows installation or upgrade.

Silent installation

To install the XenServer VM Tools for Windows silently and to prevent the system from rebooting, run one of the following commands:

```
Msiexec.exe /package managementagentx64.msi /quiet /norestart
<!--NeedCopy-->
```

Or

```
Setup.exe /quiet /norestart
<!--NeedCopy-->
```

A non-interactive, but non-silent installation can be obtained by running:

```
Msiexec.exe managementagentx64.msi /passive
<!--NeedCopy-->
```

Or

```
Setup.exe /passive
<!--NeedCopy-->
```

> **Notes:**
>
> - The `/quiet` parameter applies to the installation dialogs only, but not to the device driver installation. When the `/quiet` parameter is specified, the device driver installation requests permission to reboot if necessary.
>   - When `/quiet /norestart` is specified, the system doesn't reboot after the entire tools installation is complete. This behavior is independent of what the user specifies in the reboot dialog.
>   - When `/quiet /forcerestart` is specified, the upgrade or installation process can trigger multiple reboots. This behavior is independent of what the user specifies in the reboot dialog.
>   - When the device driver installation requests permission to reboot, a tools installation with the `quiet` parameter specified can still be in progress. Use the Task Manager to confirm whether the installer is still running.

To customize the installation settings, use the following parameters with the silent installation commands:

Parameter	Allowed values	Default	Description
ALLOWAUTOUPDATE	YES or NO	YES	Allow automatic Management Agent updates
ALLOWDRIVERINSTALL	YES or NO	YES	Install the I/O Drivers now
ALLOWDRIVERUPDATE	YES or NO	NO	Allow the automatic Management Agent updates to install updated drivers
IDENTIFYAUTOUPDATE	YES or NO	YES	Send anonymous usage information to us

For example, to do a silent install of the tools that does not allow future automatic Management Agent updates and does not send anonymous information to Cloud Software Group, Inc., run one of the following commands:

```
Msiexec.exe /package managementagentx64.msi ALLOWAUTOUPDATE=NO
   IDENTIFYAUTOUPDATE=NO /quiet /norestart
<!--NeedCopy-->
```

For interactive, silent, and passive installations, following the next system restart there might be several automated reboots before the XenServer VM Tools for Windows are fully installed. This behavior is also the case for installations with the `/norestart` flag specified. However, for installations where the `/norestart` flag is provided, the initial restart might be manually initiated.

Update XenServer VM Tools

Ensure that your XenServer VM Tools for Windows are regularly updated to the latest version, both in your VMs and in any templates that you use to create your VMs. We recommend that you snapshot your VM before updating the XenServer VM Tools. For more information about the latest version of the tools, see What's new.

> **Important:**
>
> Ensure that all requested VM restarts are completed as part of the update. Multiple restarts might be required. If all requested restarts are not completed, the VM might show unexpected behavior.

XenServer provides automatic update mechanisms for each of its components:

- The Management Agent can automatically update itself
- The I/O drivers can be updated either by the Management Agent or through Windows Update.

Alternatively, you can update either or both of these components manually.

The update method to choose for each component of the tools can depend on your environment.

Recommended update settings

For most use cases, we recommend using the following settings for updating the different components of the XenServer VM Tools for Windows:

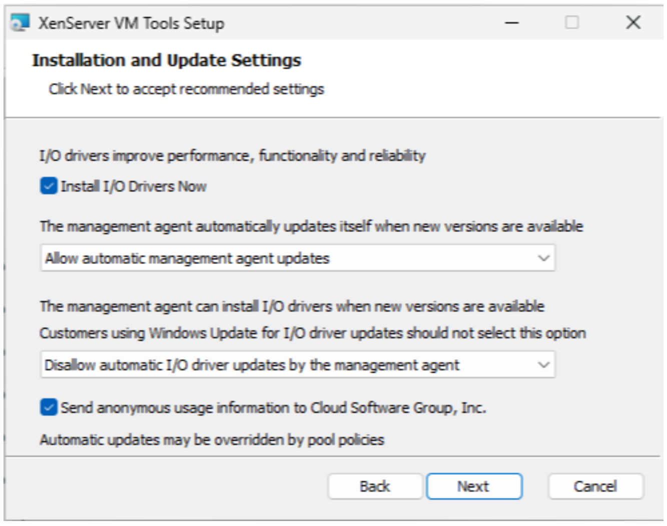
1. Enable Management Agent updates.
2. Stop the Management Agent from updating the I/O drivers.
3. Set the value of the following registry key to a REG_DWORD value of '3': `HLKM\System\CurrentControlSet\services\xenbus_monitor\Parameters\Autoreboot`.

   For more information, see ### Automatic reboots.

4. Set the I/O drivers to update through Windows Update.

During installation:

The first two settings are the default when you run the installer:


XenServer VM Tools Setup

Installation and Update Settings

Click Next to accept recommended settings

I/O drivers improve performance, functionality and reliability

☑ Install I/O Drivers Now

The management agent automatically updates itself when new versions are available

Allow automatic management agent updates

The management agent can install I/O drivers when new versions are available
Customers using Windows Update for I/O driver updates should not select this option

Disallow automatic I/O driver updates by the management agent

☑ Send anonymous usage information to Cloud Software Group, Inc.

Automatic updates may be overridden by pool policies

Settings for Citrix Provisioning targets or reset-on-boot machines

If you intend to use your Windows VM as a Citrix Provisioning target or with the reset-on-boot flag set, you cannot use any of the automated update mechanisms. We recommend that you set the following configuration on the master template that you use to create these VMs:

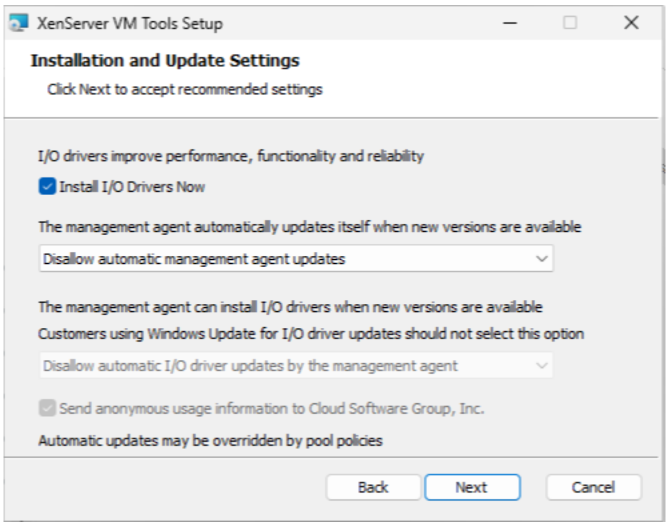
1. During creation of your VM, ensure that the `has-vendor-device` flag is set to **`false`**.

   For more information, see Create a Windows VM by using the CLI.

2. Disable Management Agent updates.

During installation:

Specify this configuration when first installing the XenServer VM Tools:


XenServer VM Tools Setup

Installation and Update Settings

Click Next to accept recommended settings

I/O drivers improve performance, functionality and reliability

[✓] Install I/O Drivers Now

The management agent automatically updates itself when new versions are available

Disallow automatic management agent updates

The management agent can install I/O drivers when new versions are available
Customers using Windows Update for I/O driver updates should not select this option

Disallow automatic I/O driver updates by the management agent

[✓] Send anonymous usage information to Cloud Software Group, Inc.

Automatic updates may be overridden by pool policies

Settings for automatic updates by the Management Agent only

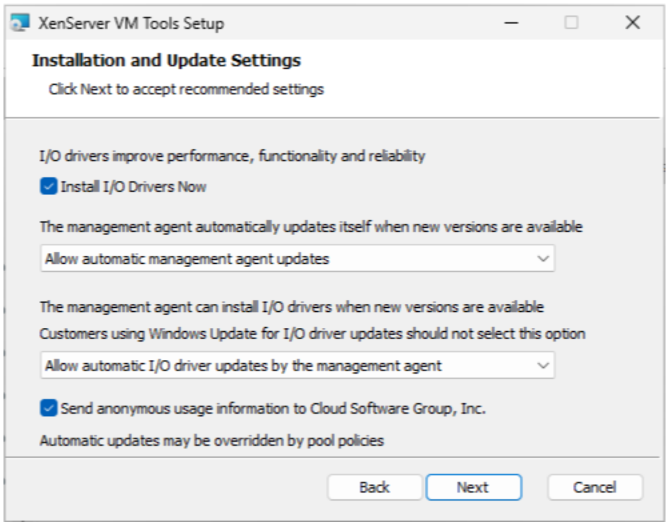
You can configure the Management Agent to update both itself and the I/O drivers. If you use this configuration, ensure that the VMs are prevented from updating the I/O drivers through Windows Update. If both mechanisms attempt to update the I/O drivers, this can result in unnecessary updates.

Choose this approach if your organization requires that you review any updates before applying them to your Windows VMs. If this is the case, you must also redirect the Management Agent to get its updates from an internal server.

1. Disable update of the I/O drivers through Windows Update.
2. Set the Management Agent to update the I/O drivers.
3. (Optional) Redirect the Management Agent updates.

During installation:

Specify this configuration when first installing the XenServer VM Tools:



Installation and Update Settings

Click Next to accept recommended settings

I/O drivers improve performance, functionality and reliability

[✓] Install I/O Drivers Now

The management agent automatically updates itself when new versions are available

Allow automatic management agent updates

The management agent can install I/O drivers when new versions are available

Customers using Windows Update for I/O driver updates should not select this option

Allow automatic I/O driver updates by the management agent

[✓] Send anonymous usage information to Cloud Software Group, Inc.

Automatic updates may be overridden by pool policies

Update the Management Agent

XenServer enables you to update the Management Agent automatically on both new and existing Windows VMs. By default, XenServer allows the automatic updating of the Management Agent. However, it does not allow the Management Agent to update the I/O drivers automatically. You can customize the Management Agent update settings during XenServer VM Tools for Windows installation. The automatic updating of the Management Agent occurs seamlessly, and does not reboot your VM. In scenarios where a VM reboot is required, a message appears on the Console tab of the VM notifying users about the required action.

You can get the Management Agent updates automatically, provided the Windows VM has access to the internet.

Manage Automatic Updates by using the CLI XenServer enables you to use the command line to manage the automatic updating of the I/O drivers and the Management Agent. You can run `msiexec.exe` with the arguments listed in the following table to specify whether the I/O drivers and the Management Agent are automatically updated. For information about installing XenServer VM Tools for Windows by using `msiexec.exe`, see Silent installation.

> **Note:**
>
> For VMs managed using either PVS or MCS, automated updates are turned off automatically when the Citrix Virtual Desktops VDA is present and it reports that the machine is non-persistent.

Argument	Values	Description
ALLOWAUTOUPDATE	YES/NO	Allow/disallow auto updating of the Management Agent
ALLOWDRIVERINSTALL	YES/NO	Allow/disallow the XenServer VM Tools for Windows installer to install I/O drivers
ALLOWDRIVERUPDATE	YES/NO	Allow/disallow the Management Agent to update the I/O drivers automatically
IDENTIFYAUTOUPDATE	YES/NO	Allow/disallow the auto update mechanism to send anonymous usage information to us

For example:

```
setup.exe  /passive /forcerestart ALLOWAUTOUPDATE=YES
    ALLOWDRIVERINSTALL=NO \
      ALLOWDRIVERUPDATE=NO IDENTIFYAUTOUPDATE=YES
<!--NeedCopy-->
```

Enable Management Agent updates To enable automatic updating of the Management Agent on a per-VM basis:

1. On the VM, open a command prompt as an administrator.
2. Run the following command:

   ```
   reg.exe ADD HKLM\SOFTWARE\XenServer\XenTools\AutoUpdate /t
       REG_DWORD /v DisableAutoUpdate /d 0
   <!--NeedCopy-->
   ```

3. Ensure that Management Agent updates are allowed by your pool. In the host console, run the following command:

   ```
   xe pool-param-set uuid=pooluuid guest-agent-config:
       auto_update_enabled=true
   <!--NeedCopy-->
   ```

Disable Management Agent updates To disable automatic updating of the Management Agent on a per-VM basis:

1. On the VM, open a command prompt as an administrator.

2. Run the following command:

```
reg.exe ADD HKLM\SOFTWARE\XenServer\XenTools\AutoUpdate /t
    REG_DWORD /v DisableAutoUpdate /d 1
<!--NeedCopy-->
```

To disable automatic updating of the Management Agent on a per-pool basis, run the following command i nthe host console:

```
xe pool-param-set uuid=pooluuid guest-agent-config:auto_update_enabled=
    false
<!--NeedCopy-->
```

Redirect the Management Agent updates XenServer enables customers to redirect Management Agent updates to an internal web server before they are installed. This redirection allows customers to review the updates before they are automatically installed on the VM.

The Management Agent uses an updates file to get information about the available updates. The name of this updates file depends on the version of the Management Agent that you use:

- For Management Agent 9.2.1.35 and later use https://pvupdates.vmd.citrix.com/autoupdate.v1.json.
- For Management Agent 9.0.0.0 to 9.2.0.27 https://pvupdates.vmd.citrix.com/updates.v9.json.

Complete the following steps to redirect the Management Agent updates:

1. Download the updates file.
2. Download the Management Agent MSI files referenced in the updates file.
3. Upload the MSI files to an internal web server that your VMs can access.
4. Update the updates file to point to the MSI files on the internal web server.
5. Upload the updates file to the web server.

Automatic updates can also be redirected on a per-VM or a per-pool basis. To redirect updates on a per-VM basis:

1. On the VM, open a command prompt as an administrator.
2. Run the command

```
reg.exe ADD HKLM\SOFTWARE\XenServer\XenTools /t REG_SZ /v
    update_url /d \
     url of the update file on the web server
<!--NeedCopy-->
```

To redirect automatic updating of the Management Agent on a per-pool basis, run the following command:

```
xe pool-param-set uuid=pooluuid guest-agent-config:auto_update_url=url
    of the update file on the web server
<!--NeedCopy-->
```

Update the I/O drivers

You can update the I/O drivers through Windows Update or by using the Management Agent. You can also turn off automatic updates and manage updates to the I/O drivers manually.

Each of the I/O drivers (xennet, xenvif, xenvbd, xeniface, and xenbus) has its own version. For information about the latest versions, see What's New.

Set the I/O drivers to update through Windows Update You can get I/O driver updates automatically from Microsoft Windows Update, provided:

- Windows Update is enabled within the VM
- The VM has access to the internet, or it can connect to a WSUS proxy server
- You are not running the Core version of Windows Server. Windows Server Core does not support using Windows Update to install or update the I/O drivers.

The **Virtualization state** section on a VM's **General** tab in XenCenter specifies whether the VM can receive updates from Windows Update. The mechanism to receive I/O driver updates from Windows Update is turned on by default. If you do not want to receive I/O driver updates from Windows Update, disable Windows Update on your VM, or specify a group policy.

Disable update of the I/O drivers through Windows Update The **Virtualization state** section on a VM's **General** tab in XenCenter specifies whether the VM can receive updates from Windows Update. The mechanism to receive I/O driver updates from Windows Update is turned on by default.

For Windows VMs that already exist, if you do not want to receive I/O driver updates from Windows Update, specify a group policy.

For new Windows VMs, you can set a flag in the VM during VM creation to prevent I/O driver updates from Windows Update. For more information, see Settings for Citrix Provisioning targets or reset-on-boot machines and Create a Windows VM by using the CLI.

Set the Management Agent to update the I/O drivers During the XenServer VM Tools for Windows installation, you can specify for the Management Agent to update the I/O drivers automatically. If you prefer to update this setting after completing the XenServer VM Tools for Windows installation process, perform the following steps:

1. On the VM, open a command prompt as an administrator.
2. Run the following command:

```
reg.exe ADD HKLM\SOFTWARE\XenServer\XenTools\AutoUpdate /t REG_SZ
    /v \
    InstallDrivers /d YES
<!--NeedCopy-->
```

Stop the Management Agent from updating the I/O drivers To prevent the Management Agent from updating the I/O drivers, perform the following steps:

1. On the VM, open a command prompt as an administrator.
2. Run the following command:

```
reg.exe ADD HKLM\SOFTWARE\XenServer\XenTools\AutoUpdate /t REG_SZ
    /v \
    InstallDrivers /d NO
<!--NeedCopy-->
```

Automatic reboots

Ensure that all requested VM restarts are completed as part of the update. Multiple restarts might be required. If all requested restarts are not completed, you might see unexpected behavior.

You can set a registry key that specifies the maximum number of automatic reboots that are performed when you install the drivers through Device Manager or Windows Update. After you have installed the xenbus driver version 9.1.1.8 or later, the XenServer VM Tools for Windows use the guidance provided by this registry key.

To use this feature, we recommend that you set the following registry key as soon as possible: `HLKM\System\CurrentControlSet\services\xenbus_monitor\Parameters\Autoreboot`. The value of the registry key must be a positive integer. We recommend that you set the number of reboots in the registry key to 3.

When this registry key is set, the XenServer VM Tools for Windows perform as many reboots as are needed to complete the updates or the number of reboots specified by the registry key - whichever value is lower.

Before each reboot, Windows can display an alert for 60 seconds that warns of the upcoming reboot. You can dismiss the alert, but this action does not cancel the reboot. Because of this delay between the reboots, wait a few minutes after the initial reboot for the reboot cycle to complete.

> **Notes:**
>
> This setting is required for headless servers with static IP addresses.
>
> This automatic reboot feature only applies to updates to the Windows I/O drivers through Device Manager or Windows Update. If you are using the Management Agent installer to deploy your drivers, the installer disregards this registry key and manages the VM reboots according to its own settings.

Other configuration and queries

Find the I/O driver version

To find out the version of the I/O drivers installed on the VM:

1. Navigate to `C:\Windows\System32\drivers`.
2. Locate the driver from the list.
3. Right-click the driver and select **Properties** and then **Details**.

   The **File version** field displays the version of the driver installed on the VM.

Find the Management Agent version

To find out the version of the Management Agent installed on the VM:

1. Navigate to `C:\Program Files\XenServer\XenTools`.
2. Right-click `XenGuestAgent` from the list and click **Properties** and then **Details**.

   The **File version** field displays the version of the Management Agent installed on the VM.

Configure anonymous usage information

During the XenServer VM Tools for Windows installation, you can specify whether you would like to send anonymous usage information to Cloud Software Group, Inc. If you prefer to update this setting after completing the XenServer VM Tools for Windows installation process, perform the following steps:

1. On the VM, open a command prompt as an administrator.
2. Run the following command:

```
reg.exe ADD HKLM\SOFTWARE\XenServer\XenTools\AutoUpdate REG_SZ /v
    \
    IDENTIFYAUTOUPDATE /d YES/NO
<!--NeedCopy-->
```

Uninstall XenServer VM Tools

We don't recommend removing the XenServer VM Tools from your Windows VMs. These tools are required for your Windows VMs to be fully supported. Removing them can cause unexpected behavior. Manually uninstall your XenServer VM Tools only as a last resort.

Standard uninstall

To do a standard uninstall of the XenServer VM Tools, you can use the Windows **Add or Remove Programs** feature:

1. Create a snapshot of the VM before you start.
2. In the Windows VM, go to **Add or Remove Programs**.
3. Select **XenServer VM Tools** and click **Uninstall**.
4. Reboot the VM.

The `uninstall.exe` command

Uninstalling the XenServer VM Tools by using the Windows **Add or Remove Programs** feature calls the `<tools-install-directory>\uninstall.exe` file to perform the uninstall actions. You can instead choose to call this command from a PowerShell terminal or a command prompt with administrator privileges.

1. Create a snapshot of the VM before you start.
2. As an administrator, open a command prompt or PowerShell terminal.
3. Run the command `<tools-install-directory>\uninstall.exe`.
4. Reboot the VM.

Command options The `uninstall.exe` command accepts the following parameters:

- `help` - Displays usage information for the command.
- `log` - Generates a log file that indicates what the command has done.
- `verbose` - Prints to the console what the command has done.
- `disable` - Disables drivers that were installed by the installer MSI.
- `force-disable` - Disables the drivers in all situations.

- `hidden` - Deletes hidden devices. These devices are unused and have been superseded, but might have left stale registry entries.
- `cleanup` - Removes old uninstallers from **Add or Remove Programs**. These uninstallers can include duplicate entries from older versions of the tools.
- `purge` - (9.3.1 and later) Resets the VM to a clean state as it was before any part of the XenServer VM Tools were installed. For more information, see Full uninstall of all XenServer VM Tools components.
- `install` - (9.3.1 and later) Installs the current set of I/O drivers and prompts for a VM reboot when required.
- `reboot`- Reboots the VM after all other command operations have completed.

Full uninstall of all XenServer VM Tools components

The latest version of XenServer VM Tools for Windows (9.3.1 and later) includes the command `uninstall.exe purge`. The `purge` option on the `uninstall.exe` application resets a VM to the state before any of the I/O drivers were installed. If you are experiencing issues when upgrading your tools to a newer version or need a clean state to install a later set of tools on your VM, use this utility.

1. Create a snapshot of the VM before you start.
2. As an administrator, open a command prompt or PowerShell terminal.
3. Run the command `<tools-install-directory>\uninstall.exe purge verbose`
4. Reboot the VM.

After using this command, you do not need to perform any manual cleanup steps like you might have had to with previous versions of the XenServer VM Tools. All changes related to the XenServer VM Tools have been removed.

What does the `purge` option remove? If you use the command `uninstall.exe purge`, all traces of the XenServer VM Tools are removed from your Windows VM. The list of actions taken by this command are as follows:

- Services:
  - Disables all XenServer VM Tools services, which prevents installed drivers and services from starting on reboot.
  - Stops any running XenServer VM Tools services.
- Drivers:
  - Uninstalls I/O drivers from all device nodes.

    - Uninstalls hidden devices. This action is the same as that performed the `hidden` command line option.
    - Uninstalls cached driver packages, which removes them from the driver store. As a result, the I/O drivers are not automatically reinstalled.

- Registry:
    - Removes stale registry information used by out of support versions of the drivers.
    - Deletes tools-related keys from `HKLM\System\CurrentControlSet\Control\Class\...`
    - Deletes tools-related keys from `HKLM\System\CurrentControlSet\Services`.
    - Deletes tools-related keys from `HKLM\System\CurrentControlSet\Enum\...`
- Files:
    - Deletes any XenServer VM Tools driver files from `C:\Windows\System32` and `C:\Windows\System32\drivers`.
    - Deletes XenServer VM Tools `INF` files from `C:\Windows\INF`.
    - Deletes any stale files left by out of support versions of the tools from `C:\Program Files\Citrix\XenTools` and `C:\Program Files\XenServer\XenTools`.
- Other:
    - Deletes old entires in **Add or Remove Programs**. This action is the same as that performed the `cleanup` command line option.
    - Clears some of the InstallAgent's stale state information.
    - Removes `xenfilt.sys` from upper-filters. This change prevents `xenfilt.sys` from loading on any driver nodes.
    - Removes the `unplug` keys, which causes the VM to revert to emulated devices on reboot.
    - Removes StorNvme's StartOverride. This change forces `stornvme.sys` to start on boot and allows emulated NVMe (UEFI) boot devices to function.

What's new

The version of the XenServer VM Tools for Windows is updated independently of the version of XenServer. Ensure that your XenServer VM Tools for Windows are regularly updated to the latest version, both in your VMs and in any templates that you use to create your VMs.

The latest version of the XenServer VM Tools for Windows is available from the XenServer Downloads page.

XenServer VM Tools for Windows 9.3.2

Released Nov 27, 2023

This release rebrands the Citrix VM Tools to XenServer VM Tools.

This set of tools contains the following component versions:

- Installer: 9.3.2
- Management Agent: 9.2.2.435
- xenbus: 9.1.7.80
- xeniface: 9.1.8.69
- xennet: 9.1.5.51
- xenvbd: 9.1.6.58
- xenvif: 9.1.10.83

Improvements in 9.3.2 This release also contains the following improvements:

- Improvements to the `uninstall.exe` command.
- Changes to enable some Windows VMs to use up to 64 vCPUs where your version of XenServer and the Windows operating system support it.

Fixed issues in 9.3.2 This release includes a fix for the following issue:

- Sometimes, when installing the XenServer VM Tools, a non-fatal error can cause the installation to fail.

Earlier releases

9.3.1 Released Jan 25, 2023

This set of tools contains the following component versions:

- Installer: 9.3.1
- Management Agent: 9.2.1.35
- xenbus: 9.1.5.54
- xeniface: 9.1.5.42
- xennet: 9.1.3.34
- xenvbd: 9.1.4.37
- xenvif: 9.1.8.58

This release includes the following improvements:

- Improvements to the `uninstall.exe` utility, including the `purge` parameter. For more information, see Uninstall XenServer VM Tools.
- General improvements to the XenServer VM Tools installer.
- General improvements to string handling of registry keys.

This release contains fixes for the following issues:

- Sometimes, when the XenServer VM Tools are updated through Windows Update, the static IP settings are lost and the network settings change to use DHCP.
- On Windows VMs, the grant tables can easily become exhausted. When this occurs, read and write requests can fail or additional VIFs are not enabled correctly and fail to start.
- On rare occasions, when upgrading the XenServer VM Tools for Windows, the existing Management Agent can fail to shut down and prevent the upgrade from succeeding.
- On a Windows VM, you might see both the previous and a later version of the tools or Management Agent listed in your Installed Programs.
  - (PREVIOUS) Citrix XenServer Windows Management Agent
  - (LATER) Citrix Hypervisor PV Tools.

  After you update to the latest version of the tools, neither of these earlier names are listed. Only XenServer VM Tools is listed in your Installed Programs.

9.3.0 Released Jul 26, 2022

This set of tools contains the following component versions:

- Installer: 9.3.0
- Management Agent: 9.2.0.27
- xenbus: 9.1.4.49
- xeniface: 9.1.4.34
- xennet: 9.1.3.34
- xenvbd: 9.1.3.33
- xenvif: 9.1.6.52

This release includes the following improvements:

- General improvements to the XenServer VM Tools installer.

This release contains fixes for the following issues:

- Security software was blocking secondary disks that are marked as removable from being exposed to the OS, as a data-exfiltration prevention measure. This update enables you to flag a VBD as non-removable and have this correctly exposed through the OS.
- On a Windows VM, sometimes the IP address of an SR-IOV VIF is not visible in XenCenter.

9.2.3 Released Apr 28, 2022

This set of tools contains the following component versions:

- Installer: 9.2.3
- Management Agent: 9.1.1.13
- xenbus: 9.1.3.30
- xeniface: 9.1.4.34
- xennet:
  - 9.1.1.8 (for Windows Server 2012 and Windows Server 2012 R2)
  - 9.1.2.23 (for all other supported Windows operating systems)
- xenvbd: 9.1.2.20
- xenvif: 9.1.5.48

This release contains fixes for the following issues:

- In XenServer VM Tools for Windows version 9.2.2, time sync options are not available.
- A race condition can cause Windows VMs to show a blue screen error after live migration on Citrix Hypervisor 8.2 Cumulative Update 1.
- Windows VMs that have version 9.2.1 or 9.2.2 of the XenServer VM Tools installed and that are PVS targets can sometimes freeze with a black screen. The message "Guest Rx stalled" is present in the dom0 kernel logs. This issue more often occurs on pool coordinators than on other pool members.
- On Windows VMs with more than 8 vCPUs, Receive Side Scaling might not work because the xenvif driver fails to set up the indirection table.

9.2.2 Released Jan 14, 2022

This set of tools contains the following component versions:

- Installer: 9.2.2
- Management Agent: 9.1.1.13
- xenbus: 9.1.3.30
- xeniface: 9.1.2.22
- xennet:
  - 9.1.1.8 (for Windows Server 2012 and Windows Server 2012 R2)
  - 9.1.2.23 (for all other supported Windows operating systems)
- xenvbd: 9.1.2.20
- xenvif: 9.1.3.31

This release contains fixes for the following issues:

- During an update of the tools, the xenbus driver can prompt a reboot before driver installation is complete. Accepting the reboot can cause a blue screen error in your Windows VM.

- When compressing collected diagnostic information, the xt-bugtool diagnostics tool times out after 20s. This behavior can result in the diagnostics zip file not being correctly created.
- VNC clipboard sharing doesn't work.
- The previous versions of the drivers were not released through Windows Update.

9.2.1 Released Jun 24, 2021

This set of tools contains the following component versions:

- Installer: 9.2.1
- Management Agent: 9.1.0.10
- xenbus: 9.1.2.14
- xeniface: 9.1.1.11
- xennet: 9.1.1.8
- xenvbd: 9.1.1.8
- xenvif: 9.1.2.16

> **Note:**
>
> This set of drivers was not provided through Windows Update.

This release contains fixes for the following issues:

- In some cases, the Laptop/Slate state of the VM cannot be changed.
- After a VM is restarted it can sometimes begin to send excessive log messages to the daemon.log file.
- A race condition in driver load dependencies after an OS upgrade can prevent the XenServer VM Tools from being upgraded.
- A storage error can cause Windows VMs to crash.
- Sometimes the IP address of an SR-IOV VIF is not visible in XenCenter. To fix the issue, restart the Management Agent from within the VM's Service Manager.
- Under high network and system load, and low resources, VMs can experience bugchecks in both Citrix and third party drivers, typically with the code IRQL_NOT_LESS_OR_EQUAL. This fix improves network buffering to prevent these bugchecks.
- Upgrading the Windows I/O drivers can cause UEFI VMs to fail to boot, reporting "0xC000000E. A required device isn't connected or can't be accessed.'"'
- An issue can occur when installing the XenServer VM Tools after uninstalling a previous version of the XenServer VM Tools that returns the following error message: "This Device cannot start (code 10) (Operation failed) The requested operation was unsuccessful".

Linux VMs

March 18, 2024

When you want to create a Linux VM, create the VM using a template for the operating system you want to run on the VM. You can use a template that XenServer provides for your operating system, or one that you created previously. You can create the VM from either XenCenter or the CLI. This section focuses on using the CLI.

> **Note:**
>
> To create a VM of a newer minor update of a RHEL release than is supported for installation by XenServer, complete the following steps:
>
> - Install from the latest supported media
> - Use `yum update` to bring the VM up-to-date
>
> This process also applies to RHEL derivatives such as CentOS and Oracle Linux.

We recommend that you install the XenServer VM Tools for Linux immediately after installing the operating system. For more information, see Install XenServer VM Tools for Linux.

The overview for creating a Linux VM is as following:

1. Create the VM for your target operating system using XenCenter or the CLI.
2. Install the operating system using vendor installation media.
3. Install the XenServer VM Tools for Linux (recommended).
4. Configure the correct time and time zone on the VM and VNC as you would in a normal non-virtual environment.

XenServer supports the installation of many Linux distributions as VMs.

> **Warning:**
>
> The **Other install media** template is for advanced users who want to attempt to install VMs running unsupported operating systems. XenServer has been tested running only the supported distributions and specific versions covered by the standard supplied templates. Any VMs installed using the **Other install media** template are *not* supported.

For information regarding specific Linux distributions, see Installation notes for Linux distributions.

Supported Linux distributions

For a list of supported Linux distributions, see Guest operating system support.

Other Linux distributions are **not** supported. However, distributions that use the same installation mechanism as Red Hat Enterprise Linux (for example, Fedora Core) might be successfully installed using the same template.

Create a Linux VM

This section includes procedures for creating a Linux VM by installing the OS from a physical CD/DVD or from a network-accessible ISO.

Create a Linux VM by using the xe CLI

This section shows the CLI procedure for creating a Linux VM by installing the OS from a physical CD/DVD or from a network-accessible ISO.

1. Create a VM from the appropriate template. The UUID of the VM is returned:

   ```
   xe vm-install template=template-name new-name-label=vm-name
   <!--NeedCopy-->
   ```

2. (Optional) Change the boot mode of the VM.

   ```
   xe vm-param-set uuid=<uuid> HVM-boot-params:firmware=<mode>
   xe vm-param-set uuid=<UUID> platform:device-model=qemu-upstream-
       uefi
   xe vm-param-set uuid=<uuid> platform:secureboot=<option>
   <!--NeedCopy-->
   ```

   Tha value of `mode` can be either `BIOS` or `uefi` and defaults to `uefi` if that option is supported for your VM operating system. Otherwise, the mode defaults to `BIOS`. The value of `option` can be set to either **`true`** or **`false`**. If you do not specify the Secure Boot option, it defaults to `auto`.

   For more information, see Guest UEFI boot and Secure Boot.

3. Add a virtual CD-ROM to the new VM:

   - If you are installing from a CD or DVD, get the name of the physical CD drive on the XenServer host:

     ```
     xe cd-list
     <!--NeedCopy-->
     ```

     The result of this command gives you something like SCSI 0:0:0:0 for the `name-label` field.

     Use this value parameter as the `cd-name` parameter:

```
xe vm-cd-add vm=vm_name cd-name="host_cd_drive_name_label"
    device=3
<!--NeedCopy-->
```

- If you are installing from a network-accessible ISO, use the name of the ISO from the ISO library-label as the value for the `cd-name` parameter:

```
xe vm-cd-add vm=vm_name cd-name="iso_name.iso" device=3
<!--NeedCopy-->
```

4. Insert the operating system installation CD into the CD drive on the XenServer host.
5. Open a console to the VM with XenCenter or an SSH terminal and follow the steps to perform the OS installation.
6. Start the VM. It boots straight into the operating system installer:

```
xe vm-start uuid=UUID
<!--NeedCopy-->
```

7. Install the guest utilities and configure graphical display. For more information, see Install the XenServer VM Tools for Linux.

Create a Linux VM by using XenCenter

1. On the XenCenter toolbar, click the **New VM** button to open the New VM wizard.

   The New VM wizard allows you to configure the new VM, adjusting various parameters for CPU, storage, and networking resources.

2. Select a VM template and click **Next**.

   Each template contains the setup information that is required to create a VM with a specific guest operating system (OS), and with optimum storage. This list reflects the templates that XenServer currently supports.

   > **Note:**
   >
   > If the OS that you are installing on your VM is compatible only with the original hardware, check the **Copy host BIOS strings to VM** box. For example, you might use this option for an OS installation CD that was packaged with a specific computer.
   >
   > After you first start a VM, you cannot change its BIOS strings. Ensure that the BIOS strings are correct before starting the VM for the first time.

   To copy BIOS strings using the CLI, see Install VMs from Reseller Option Kit (BIOS-locked) Media.

   Advanced users can set user-defined BIOS strings. For more information, see User-defined BIOS strings.

3. Enter a name and an optional description for the new VM.
4. Choose the source of the OS media to install on the new VM.

   Installing from a CD/DVD is the simplest option for getting started.

   a) Choose the default installation source option (DVD drive)
   b) Insert the disk into the DVD drive of the XenServer host

   XenServer also allows you to pull OS installation media from a range of sources, including a pre-existing ISO library.

   To attach a pre-existing ISO library, click **New ISO library** and indicate the location and type of the ISO library. You can then choose the specific operating system ISO media from the list.

5. In the **Installation Media** tab, you can choose a boot mode for the VM. By default, XenCenter selects the most secure boot mode available for the VM operating system version.

   > **Notes:**
   >
   > - The **UEFI Boot** and **UEFI Secure Boot** options appear grayed out if the VM template you have chosen does not support UEFI boot.
   > - You cannot change the boot mode after you boot the VM for the first time.
   >
   > For more information, see Guest UEFI boot and Secure Boot.

6. Select a home server for the VM.

   A home server is the host which provides the resources for a VM in a pool. When you nominate a home server for a VM, XenServer attempts to start the VM on that host. If this action is not possible, an alternate host within the same pool is selected automatically. To choose a home server, click **Place the VM on this server** and select a host from the list.

   > **Notes:**
   >
   > - In WLB-enabled pools, the nominated home server isn't used for starting, restarting, resuming, or migrating the VM. Instead, Workload Balancing nominates the best host for the VM by analyzing XenServer resource pool metrics and by recommending optimizations.
   > - If a VM has one or more virtual GPUs assigned to it, the home server nomination doesn't take effect. Instead, the host nomination is based on the virtual GPU placement policy set by the user.
   > - During rolling pool upgrade, the home server is not considered when migrating the VM. Instead, the VM is migrated back to the host it was on before the upgrade.
   >   If you do not want to nominate a home server, click **Don't assign this VM a home server**. The VM is started on any host with the necessary resources.

Click **Next** to continue.

7. Allocate processor and memory resources for the VM. Click **Next** to continue.
8. Assign a virtual GPU.

   If vGPU is supported, the New VM wizard prompts you to assign a dedicated GPU or one or more virtual GPUs to the VM. This option enables the VM to use the processing power of the GPU. With this feature, you have better support for high-end 3D professional graphics applications such as CAD/CAM, GIS, and Medical Imaging applications.
9. Allocate and configure storage for the new VM.

   Click **Next** to select the default allocation (24 GB) and configuration, or you might want to do the following extra configuration:

   - Change the name, description, or size of your virtual disk by clicking **Edit**.
   - Add a new virtual disk by selecting **Add**.
10. Configure networking on the new VM.

    Click **Next** to select the default NIC and configurations, including an automatically created unique MAC address for each NIC. Alternatively, you might want to do the following extra configuration:

    - Change the physical network, MAC address, or Quality of Service (QoS) priority of the virtual disk by clicking **Edit**.
    - Add a new virtual NIC by selecting **Add**.
11. Review settings, and then click **Create Now** to create the VM and return to the **Search** tab.

    An icon for your new VM appears under the host in the **Resources** pane.

    On the **Resources** pane, select the VM, and then click the **Console** tab to see the VM console.
12. Follow the OS installation screens and make your selections.
13. After the OS installation completes and the VM reboots, install the XenServer VM Tools for Linux.

Create a Linux VM by using PXE boot

You can use PXE boot to install the operating system of your Linux VM. This approach can be useful when you have to create many Linux VMs.

To install by using PXE boot, set up the following prerequisites in the network where your Linux VMs are located:

- DHCP server that is configured to direct any PXE boot installation requests to the TFTP server
- TFTP server that hosts the installation files for the Linux operating system

When creating the Linux VM, run the following commands:

1. Create a VM from the appropriate template. The UUID of the VM is returned:

   ```
   xe vm-install template=template-name new-name-label=vm-name
   <!--NeedCopy-->
   ```

2. Set the boot order to boot from the disk and then from the network:

   ```
   xe vm-param-set uuid=<UUID> HVM-boot-params:order=cn
   <!--NeedCopy-->
   ```

3. Start the VM to begin the PXE boot installation:

   ```
   xe vm-start uuid=<UUID>
   <!--NeedCopy-->
   ```

4. Install the guest utilities and configure graphical display. For more information, see Install the XenServer VM Tools for Linux.

For more information about using PXE boot to install Linux operating systems, see the operating system documentation:

- Debian: Installing Debian using network booting
- Red Hat: Starting a Kickstart installation automatically using PXE
- CentOS: PXE Setup
- SLES: Preparing Network Boot Environment
- Ubuntu: Netbooting the server installer on amd64

Install XenServer VM Tools for Linux

Although all supported Linux distributions are natively paravirtualized (and don't need special drivers for full performance), XenServer VM Tools for Linux provide a guest agent. This guest agent provides extra information about the VM to the host. Install the guest agent on each Linux VM to benefit from the following features:

- View VM performance data in XenCenter.

  For example, the following memory performance values are visible in XenCenter only when the XenServer VM Tools are installed: "Used Memory", "Disks", Network"and "Address".

- In XenCenter, view the Linux guest operating system information.

- In the XenCenter **Networking** tab, view the IP address of the VM.

- Launch an SSH Console to the VM from XenCenter.

- Adjust the number of vCPUs on a running Linux VM.

- Enable Dynamic Memory Control (DMC).

  > **Note:**
  >
  > You cannot use the Dynamic Memory Control (DMC) feature on Red Hat Enterprise Linux 8, Red Hat Enterprise Linux 9, Rocky Linux 8, Rocky Linux 9, or CentOS Stream 9 VMs as these operating systems do not support memory ballooning with the Xen hypervisor.

It is important to keep the Linux guest agent up-to-date as you upgrade your XenServer host. For more information, see Update Linux kernels and guest utilities.

> **Note:**
>
> Before installing the guest agent on a SUSE Linux Enterprise Desktop or Server 15 guest, ensure that `insserv-compat-0.1-2.15.noarch.rpm` is installed on the guest.

To install the XenServer VM Tools for Linux:

1. Download the XenServer VM Tools for Linux file from the XenServer Downloads page.
2. Copy the `LinuxGuestTools-xxx.tar.gz` file to your Linux VM or to a shared drive that the Linux VM can access.
3. Extract the contents of the tar file: `tar -xzf LinuxGuestTools-xxx.tar.gz`
4. Run the installation script as the root user:

   ```
   /<extract-directory>/install.sh
   <!--NeedCopy-->
   ```

5. If the kernel has been upgraded, or the VM was upgraded from a previous version, reboot the VM now.

Uninstall XenServer VM Tools for Linux

From version 8.4.0-1, you can use the `install.sh` script to uninstall XenServer VM Tools for Linux. To uninstall the tools, run the following command as the root user:

```
/<extract-directory>/install.sh -u
<!--NeedCopy-->
```

Install third-party drivers on your Secure Boot Linux VM

To install third-party drivers in a Linux VM that has UEFI Secure Boot enabled, you must create a signing key, add it to the VM as a machine owner key (MOK), and use that key to sign the driver. For example, if you use the XenServer graphics capabilities with your Linux VM, you might need to install the NVIDIA graphics driver on your VM.

Complete the following steps to create a key and use it to install a third-party driver:

1. Generate a public key and private key pair.
2. Enroll the public key in MOK.
3. Set the keys you created as the module signing keys for the driver.

The following example shows this procedure in detail for an NVIDIA graphics driver on a secure-boot-enabled Ubuntu VM:

1. Download the NVIDIA driver to your VM.
2. Create a directory (for example, `/root/module-signing`) to hold the keys:

   ```
   mkdir -p /root/module-signing
   ```

3. Create a public and private key to use to sign the driver:

   ```
   openssl req -new -x509 -newkey rsa:2048 -keyout /root/module-
       signing/Nvidia.key -outform DER -out /root/module-signing/
       Nvidia.der -nodes -days 36500 -subj "/CN=Graphics Drivers"
   ```

4. Import the public key into MOK by using `mokutil`:

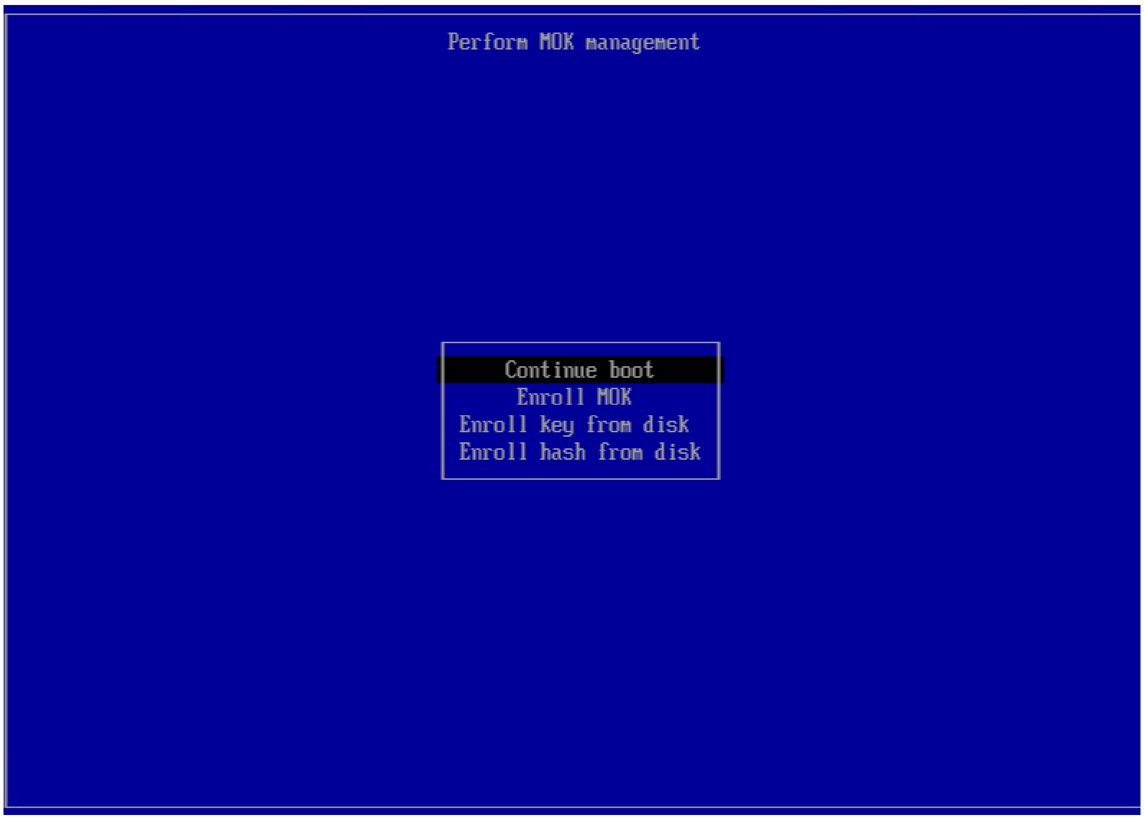
   ```
   mokutil --import /root/module-signing/Nvidia.der
   ```

   You are asked to create a password during this step. When you next boot, you are prompted to provide the password you create here.

5. Ensure that the VM boot target is set to graphical:

   ```
   systemctl set-default graphical.target
   ```

6. Reboot the VM.
7. During the boot, the **Perform MOK manageement** GUI is displayed.



In this interface, complete the following steps:

a) Select **Enroll MOK** > **Continue**.
b) When asked whether to **Enroll the key(s)?**, select **Yes**.
c) When prompted, provide the password you created when you imported the public key (Step 4).

8. Install the package `libglvnd-dev`:

```
apt install pkg-config libglvnd-dev
```

9. Install the NVIDIA driver, specifying the keys you created as the module signing keys:

```
bash ./NVIDIA-Linux-x86_64-535.129.03-grid.run --module-signing-
    secret-key=/root/module-signing/Nvidia.key --module-signing-
    public-key=/root/module-signing/Nvidia.der
```

Installation notes for Linux distributions

This section lists vendor-specific, configuration information to consider before creating the specified Linux VMs.

For more detailed release notes on all distributions, see Linux VM Release Notes.

Red Hat Enterprise Linux* 7 (32-/64-bit)

The new template for these guests specifies 2 GB RAM. This amount of RAM is a requirement for a successful install of v7.4 and later. For v7.0 - v7.3, the template specifies 2 GB RAM, but as with previous versions of XenServer, 1 GB RAM is sufficient.

> **Note:**
>
> This information applies to both Red Hat and Red Hat derivatives.

Apt repositories (Debian)

For infrequent or one-off installations, it is reasonable to use a Debian mirror directly. However, if you intend to do several VM installations, we recommend that you use a caching proxy or local mirror. Either of the following tools can be installed into a VM.

- `Apt-cacher`: An implementation of proxy server that keeps a local cache of packages
- `debmirror`: A tool that creates a partial or full mirror of a Debian repository

Prepare to clone a Linux VM

Typically, when cloning a VM or a computer, unless you generalize the cloned image, attributes unique to that machine are duplicated in your environments. Some of the unique attributes that are duplicated when cloning are the IP address, SID, or MAC address.

As a result, XenServer automatically changes some virtual hardware parameters when you clone a Linux VM. When you copy the VM using XenCenter, XenCenter automatically changes the MAC address and IP address for you. If these interfaces are configured dynamically in your environment, you might not need to modify the cloned VM. However, if the interfaces are statically configured, you might need to modify their network configurations.

The VM may need to be customized to be made aware of these changes. For instructions for specific supported Linux distributions, see Linux VM Release Notes.

Machine name

A cloned VM is another computer, and like any new computer in a network, it must have a unique name within the network domain.

IP address

A cloned VM must have a unique IP address within the network domain it is part of. Generally, this requirement is not a problem when DHCP is used to assign addresses. When the VM boots, the DHCP server assigns it an IP address. If the cloned VM had a static IP address, the clone must be given an unused IP address before being booted.

MAC address

There are two situations when we recommend disabling MAC address rules before cloning:

1. In some Linux distributions, the MAC address for the virtual network interface of a cloned VM is recorded in the network configuration files. However, when you clone a VM, XenCenter assigns the new cloned VM a different MAC address. As a result, when the new VM is started for the first time, the network does recognize the new VM and does not come up automatically.
2. Some Linux distributions use udev rules to remember the MAC address of each network interface, and persist a name for that interface. This behavior is intended so that the same physical NIC always maps to the same `eth`*n* interface, which is useful with removable NICs (like laptops). However, this behavior is problematic in the context of VMs.

   For example, consider the behavior in the following case:

   ```
   1.  Configure two virtual NICs when installing a VM
   1.  Shut down the VM
   1.  Remove the first NIC
   ```

   When the VM reboots, XenCenter shows just one NIC, but calls it `eth0`. Meanwhile the VM is deliberately forcing this NIC to be `eth1`. The result is that networking does not work.

For VMs that use persistent names, disable these rules before cloning. If you do not want to turn off persistent names, you must reconfigure networking inside the VM (in the usual way). However, the information shown in XenCenter does not match the addresses actually in your network.

Update Linux kernels and guest utilities

The Linux guest utilities can be updated by rerunning the `install.sh` script from the XenServer VM Tools for Linux (see Install the XenServer VM Tools for Linux).

For `yum`-enabled distributions, CentOS and RHEL, `xe-guest-utilities` installs a `yum` configuration file to enable subsequent updates to be done using `yum` in the standard manner.

For Debian, `/etc/apt/sources.list` is populated to enable updates using apt by default.

When upgrading, we recommend that you always rerun `install.sh`. This script automatically determines if your VM needs any updates and installs if necessary.

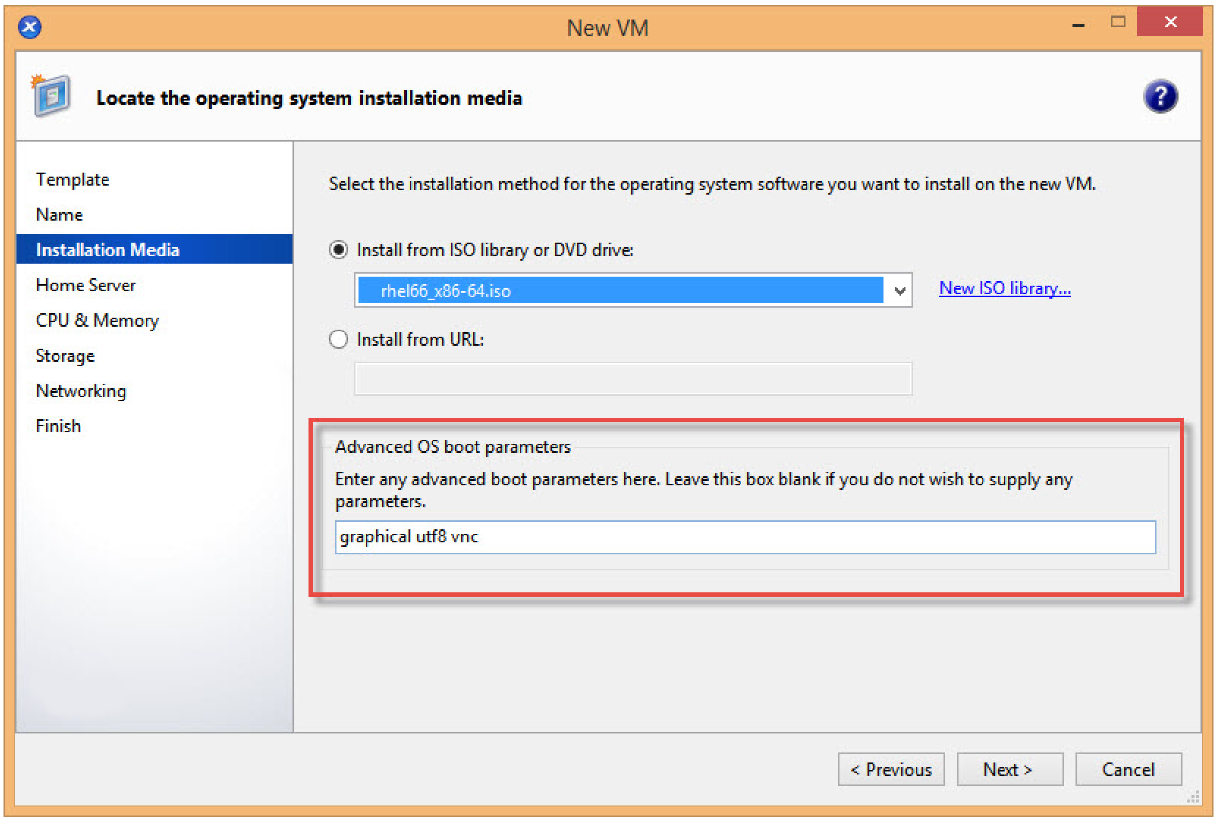
Linux VM release notes

Most modern Linux distributions support Xen paravirtualization directly, but have different installation mechanisms and some kernel limitations.

RHEL graphical install support

To use the graphical installer, in XenCenter step through the **New VM** wizard. In the **Installation Media** page, in the **Advanced OS boot parameters** section, add `vnc` to the list parameters:

```
graphical utf8 vnc
<!--NeedCopy-->
```



Select the installation method for the operating system software you want to install on the new VM.

(●) Install from ISO library or DVD drive: rhel66_x86-64.iso

( ) Install from URL:

Advanced OS boot parameters

Enter any advanced boot parameters here. Leave this box blank if you do not wish to supply any parameters.

graphical utf8 vnc

You are prompted to provide networking configuration for the new VM to enable VNC communication. Work through the remainder of the New VM wizard. When the wizard completes, in the **Infrastructure** view, select the VM, and click **Console** to view a console session of the VM. At this point, it uses the standard installer. The VM installation initially starts in text mode, and may request network configuration. Once provided, the **Switch to Graphical Console** button is displayed in the top right corner of the XenCenter window.

Red Hat Enterprise Linux 7

After migrating or suspending the VM, RHEL 7 guests might freeze during resume. For more information, see Red Hat issue 1141249.

Red Hat Enterprise Linux 8

You cannot use the Dynamic Memory Control (DMC) feature on Red Hat Enterprise Linux 8, Red Hat Enterprise Linux 9, Rocky Linux 8, Rocky Linux 9, or CentOS Stream 9 VMs as these operating systems do not support memory ballooning with the Xen hypervisor.

CentOS 7

For the list of CentOS 7 release notes, see Red Hat Enterprise Linux 7.

Oracle Linux 7

For the list of Oracle Linux 7 release notes, see Red Hat Enterprise Linux 7.

Scientific Linux 7

For the list of Scientific Linux 7 release notes, see Red Hat Enterprise Linux 7.

Debian 10

If you install Debian 10 (Buster) by using PXE network boot, do not add `console=tty0` to the boot parameters. This parameter can cause issues with the installation process. Use only `console=hvc0` in the boot parameters.

For more information, see Debian issues 944106 and 944125.

SUSE Linux Enterprise 12

Prepare a SLES guest for cloning

> **Note:**
>
> Before you prepare a SLES guest for cloning, ensure that you clear the udev configuration for network devices as follows:

```
cat< /dev/null > /etc/udev/rules.d/30-net_persistent_names.rules
```

To prepare an SLES guest for cloning:

1. Open the file `/etc/sysconfig/network/config`
2. Edit the line that reads:

   ```
   FORCE_PERSISTENT_NAMES=yes
   <!--NeedCopy-->
   ```

   To

   ```
   FORCE_PERSISTENT_NAMES=no
   <!--NeedCopy-->
   ```

3. Save the changes and reboot the VM.
   For more information, see Prepare to Clone a Linux VM.

Ubuntu 18.04 (deprecated)

Ubuntu 18.04 offers the following types of kernel:

- The General Availability (GA) kernel, which is not updated at point releases
- The Hardware Enablement (HWE) kernel, which is updated at point releases

Some minor versions of Ubuntu 18.04 (for example 18.04.2 and 18.04.3) use a HWE kernel by default that can experience issues when running the graphical console. To work around these issues, you can choose to run these minor versions of Ubuntu 18.04 with the GA kernel or to change some of the graphics settings. For more information, see CTX265663 - Ubuntu 18.04.2 VMs can fail to boot on XenServer.

VM memory

August 31, 2023

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

XenCenter provides a graphical display of memory usage in its **Memory** tab. For more information, see the XenCenter documentation.

Dynamic Memory Control (DMC) provides the following benefits:

- You can add or delete memory without restarting the VMs, providing a seamless experience to the user.
- When hosts are full, DMC allows you to start more VMs on these hosts, reducing the amount of memory allocated to the running VMs proportionally.

What is Dynamic Memory Control (DMC)?

XenServer DMC 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 host.

Without DMC, when a host is full, starting additional VMs fail with "out of memory" errors. To reduce the existing VM memory allocation and make room for more VMs, edit each VM's memory allocation and then restart the VM. When using DMC, XenServer attempts to reclaim memory by automatically reducing the current memory allocation of running VMs within their defined memory ranges. XenServer attempts to reclaim memory even when the host is full.

> **Notes:**
>
> Dynamic Memory Control is not supported with VMs that have a virtual GPU.

The concept of dynamic range

For each VM, the administrator can set a dynamic memory range. The dynamic memory range is the range within which memory can be added/removed from the VM without requiring a restart. 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, forcing XenServer to ensure that this amount of memory is allocated to the VM. If new VMs are required to start on "full" hosts, 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 operates 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 1,024 MB, it gives the VM a Dynamic Memory Range (DMR) of 512–1024 MB, within which it operates. XenServer *guarantees* always to assign each VM memory within its specified DMR when using DMC.

The concept of static range

Many operating systems that XenServer supports do not fully 'understand'the notion of dynamically adding or deleting memory. As a result, XenServer must declare the maximum amount of memory that a VM is asked to consume at the time that it restarts. Declaring the maximum amount of memory 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 when the VM is running. For a particular boot, the dynamic range is constrained such as to be always contained within this static range. The static minimum (the lower bound of the static range) protects the administrator and is set to the lowest amount of memory that the OS can run with XenServer.

> **Note:**
>
> We recommend that you do not change the static minimum level as the static minimum level is set at the supported level per operating system. See the memory constraints table for more details.
>
> Setting a static maximum level higher than a dynamic max allows you to allocate more memory to a VM in future without restarting the VM.

DMC behavior

Automatic VM squeezing

- If DMC is not enabled, when hosts are full, new VM starts fail with 'out of memory'errors.
- When DMC is enabled, even when hosts are full, XenServer attempts 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 receive their Dynamic Maximum Memory level
- When the host's memory is scarce - All running VMs 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 can cause it to restart slowly. Likewise, if you allocate too little memory to a VM, it can start slowly.
- Setting the dynamic memory minimum for a VM too low can 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 useful in virtual server environments, and in situations where you know exactly how much memory you want a guest to use. XenServer adjusts 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 selects a target from the range and adjusts the guest's memory allocation to meet the target. Specifying a dynamic range is 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 selects 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. 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 previous constraint, we support only certain guest memory configurations for each supported operating system. 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 advised to keep memory properties within the supported limits to avoid performance or stability problems.

For detailed guidelines on the minimum and maximum memory limits for each supported operating system, see Guest operating system support.

> **Warning:**
>
> When configuring guest memory, we advise 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 can lead to stability problems within your guest.
>
> The dynamic minimum must be greater than or equal to a quarter of the static maximum for all supported operating systems. Reducing the lower limit below the dynamic minimum can also lead to stability problems. Administrators are encouraged to calibrate the sizes of their VMs carefully, and ensure that their working set of applications function reliably at dynamic-minimum.
>
> The dynamic minimum must be at least 75% of the static maximum. A lower amount can cause in-guest failures and is not supported.

xe CLI commands

Display the static memory properties of a VM

1. Find the UUID of the required VM:

   ```
   xe vm-list
   <!--NeedCopy-->
   ```

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 }

   <!--NeedCopy-->
   ```

   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
   <!--NeedCopy-->
   ```

   The example shows that the static maximum memory for this VM is 268,435,456 bytes (256 MB).

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
   <!--NeedCopy-->
   ```

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 }

   <!--NeedCopy-->
   ```

   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
   <!--NeedCopy-->
   ```

   The example shows that the dynamic maximum memory for this VM is 134,217,728 bytes (128 MB).

Update memory properties

> **Warning:**
>
> 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
<!--NeedCopy-->
```

Update the dynamic memory range of a virtual machine:

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

```
<!--NeedCopy-->
```

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

```
xe vm-memory-target-set target=value vm=vm-name
<!--NeedCopy-->
```

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
<!--NeedCopy-->
```

Notes:

- To allocate a specific amount memory to a VM that doesn'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 must shut down the VM.

Update individual memory properties

Warning:

Do not change the static minimum level as it is set at the supported level per operating system. For more information, see Memory constraints.

Update the dynamic memory properties of a VM.

1. Find the UUID of the required VM:

   ```
   xe vm-list
   <!--NeedCopy-->
   ```

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
   <!--NeedCopy-->
   ```

The following example changes the dynamic maximum to 128 MB:

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

Migrate VMs

April 18, 2024

You can migrate a running VM by using live migration or storage live migration to move a VM's Virtual Disk Image (VDI) without any VM downtime.

Live migration and storage live migration

The following sections describe the compatibility requirements and limitations of live migration and storage live migration.

Live migration

Live migration is available in all versions of XenServer. This feature enables you to move a running VM from one host to another host, when the VM's disks (VDIs) are on storage shared by both hosts. Pool maintenance features such as high availability and Rolling Pool Upgrade (RPU) can automatically move VMs by using live migration. These features allow for workload leveling, infrastructure resilience, and the upgrade of server software, without any VM downtime.

During the live migration of a VM, its memory is transferred as a data stream between two hosts using the network. The migration stream compression feature compresses this data stream, speeding up the memory transfer on slow networks. This feature is disabled by default, but this can be changed by using XenCenter or the xe CLI. For more information, see Pool Properties - Advanced and Pool parameters. Alternatively, you can enable compression when migrating a VM by using the command line. For more information, see the `vm-migrate` command in VM Commands.

The parallel host evacuation feature speeds up host evacuation time (during host updates) by moving VMs off a host in parallel instead of sequentially. By default, this feature is enabled and the VMs are migrated in batches of 10 in parallel. You can change the default batch size in the `/etc/xapi.conf` file.

> **Note:**
>
> Storage can only be shared between hosts in the same pool. As a result VMs can only be migrated to hosts in the same pool.

> Intel GVT-g is not compatible with live migration, storage live migration, or VM Suspend. For information, see Graphics.

Storage live migration

> **Notes:**
>
> - Do not use storage live migration in Citrix Virtual Desktops deployments.
> - Storage live migration cannot be used on VMs that have changed block tracking enabled. Disable changed block tracking before attempting storage live migration.
> - Storage live migration cannot be used on VMs whose VDIs are on a GFS2 SR.

Storage live migration allows a VM to be moved from one host to another, when the VM's disks are not on storage shared between the two hosts. As a result, VMs stored on local storage can be migrated without downtime and VMs can be moved from one pool to another. This feature enables system administrators to:

- Rebalance VMs between XenServer pools (for example from a development environment to a production environment).
- Upgrade and update standalone XenServer hosts without any VM downtime.
- Upgrade XenServer server hardware.

> **Note:**
>
> - Migrating a VM from one host to another preserves the VM *state*. The state information includes information that defines and identifies the VM and the historical performance metrics, such as CPU and network usage.
> - To improve security, you can close TCP port 80 on the management interface of your XenServer hosts. However, you cannot migrate a VM from a Citrix Hypervisor 8.2 CU1 pool without hotfix XS82ECU1033 installed, to a XenServer pool with port 80 closed. To do so, install XS82ECU1033 on your Citrix Hypervisor 8.2 CU1 pool or temporarily open port 80 on your XenServer pool. For more information about how to close port 80, see Restrict use of port 80.

Compatibility requirements

When migrating a VM with live migration or storage live migration, the new VM and server must meet the following compatibility requirements.

General requirements:

- The target host must have the same or a more recent version of XenServer installed as the source host.
- XenServer VM Tools for Windows must be installed on each Windows VM that you want to migrate.
- You cannot concurrently migrate more than three VMs that have their source location in the same pool.

CPU requirements:

- If the CPUs on the source and target host are different, the target host must be at least as capable as the source host. Generally, this means that the target has the same or a newer CPU.
  - If you are migrating within the same pool, the pool automatically attempts to make a VM compatible.
  - If you are migrating between pools, you must ensure that the VM is compatible with the feature set in the destination pool.
- You cannot live migrate a VM between AMD and Intel processors.

Memory requirements:

- The target host must have sufficient spare memory capacity or be able to free sufficient capacity using Dynamic Memory Control. If there is not enough memory, the migration fails to complete.
- Storage migration only: A host in the source pool must have sufficient spare memory capacity to run a halted VM that is being migrated. This requirement enables the halted VM to be started at any point during the migration process.

Disk space requirements:

- Storage live migration only: The target storage must have enough free disk space available for the incoming VMs. The free space required can be three times the VDI size (without snapshots). If there is not enough space, the migration fails to complete.
- The source storage must have enough free disk space to create temporary snapshots of the VM's VDIs during the migration. If there is not enough space, the migration fails to complete. The free space required can be up to two times the size of the VM's disk.

Limitations and caveats

Live migration and storage live migration are subject to the following limitations and caveats:

- Storage live migration cannot be used with VMs created by Machine Creation Services.

- VMs using PCI pass-through devices cannot be migrated (except in the case of NVIDIA SR-IOV GPUs). For more information, see Use SR-IOV enabled NICs.
- VMs with attached vUSBs cannot be migrated.
- VMs with the parameter `no-migrate` set cannot be migrated.
- Intel GVT-g is not compatible with live migration and storage live migration. For more information, see Graphics overview.
- You cannot use storage live migration to migrate VMs that have changed block tracking enabled. Disable changed block tracking before attempting storage live migration. For more information, see Changed Block Tracking.
- VMs that have the `on-boot` option set to `reset` cannot be migrated. For more information, see Intellicache.
- If you use the high availability feature and the VM being migrated is marked as protected, you might receive a warning during live migration if the operation causes the HA constraints to not be met.
- VM performance is reduced during migration.
- Time to completion of VM migration depends on the memory footprint of the VM, and its activity. In addition, the size of the VDI and the storage activity of the VDI can affect VMs being migrated with storage live migration. VMs with vGPUs attached migrate the entire vGPU state while the VM is paused. We recommended that you use a fast network card on the management network to reduce downtime, especially with vGPUs that have large amounts of memory.
- If live migration fails, for example, in the case of a network error, the VM on the source host can instantly go to a halted state.

Migrate a VM using XenCenter

1. In the Resources pane, select the VM and do one of the following:

    - To migrate a running or suspended VM using live migration or storage live migration, on the **VM** menu, click **Migrate to Server** and then **Migrate VM wizard**. This action opens the **Migrate VM** wizard.

    - To move a stopped VM: On the **VM** menu, select **Move VM**. This action opens the **Move VM** wizard.

2. From the **Destination** list, select a standalone host or a pool.

3. From the **Home Server** list, select a host to assign as the home server for the VM and click **Next**.

4. In the **Storage** tab, specify the storage repository where you would like to place the migrated VM's virtual disks, and then click **Next**.

    - The **Place all migrated virtual disks on the same SR** radio button is selected by default and displays the default shared SR on the destination pool.

    - Click **Place migrated virtual disks onto specified SRs** to specify an SR from the **Storage Repository** list. This option allows you to select different SR for each virtual disk on the migrated VM.

5. From the **Storage network** list, select a network on the destination pool that is used for the live migration of the VM's virtual disks. Click **Next**.

   > **Note:**
   >
   > Due to performance reasons, it is recommended that you do not use your management network for live migration.

6. Review the configuration settings and click **Finish** to start migrating the VM.

If you are upgrading from 7.1 CU2 to 8.2 CU1, you might need to shut down and boot all VMs after migrating your VMs, to ensure that new virtualization features are picked up.

Live VDI migration

Live VDI migration allows the administrator to relocate the VMs Virtual Disk Image (VDI) without shutting down the VM. This feature enables administrative operations such as:

- Moving a VM from cheap local storage to fast, resilient, array-backed storage.
- Moving a VM from a development to production environment.
- Moving between tiers of storage when a VM is limited by storage capacity.
- Performing storage array upgrades.

Limitations and caveats

Live VDI Migration is subject to the following limitations and caveats

- Do not use storage live migration in Citrix Virtual Desktops deployments.
- IPv6 Linux VMs require a Linux Kernel greater than 3.0.
- If you perform live VDI migration on a VM that has a vGPU, vGPU live migration is used. The host must have enough vGPU space to make a copy of the vGPU instance on the host. If the pGPUs are fully employed, VDI migration might not be possible.
- When you do a VDI live migration for a VM that remains on the same host, that VM temporarily requires twice the amount of RAM.

To move virtual disks

1. In the **Resources** pane, select the SR where the Virtual Disk is stored and then click the **Storage** tab.
2. In the **Virtual Disks** list, select the Virtual Disk that you would like to move, and then click **Move**.
3. In the **Move Virtual Disk** dialog box, select the target SR that you would like to move the VDI to.

   > **Note:**
   >
   > Ensure that the SR has sufficient space for another virtual disk: the available space is shown in the list of available SRs.

4. Click **Move** to move the virtual disk.

Import and export VMs

April 11, 2024

XenServer allows you to import VMs from and export them to various different formats. Using the XenCenter Import wizard, you can import VMs from disk images (VHD and VMDK), Open Virtualization Format (OVF and OVA) and XenServer XVA format. You can even import VMs that have been created on other virtualization platforms, such as those offered by VMware and Microsoft.

> **Note:**
>
> When importing VMs that have been created using other virtualization platforms, configure or *fix up* the guest operating system to ensure that it boots on XenServer. The Operating System Fixup feature in XenCenter aims to provide this basic level of interoperability. For more information, see Operating system fixup.

Using the XenCenter **Export** wizard, you can export VMs to Open Virtualization Format (OVF and OVA) and XenServer XVA format.

You can also use the xe CLI to import VMs from and export them to XenServer XVA format.

Supported formats

Format	Description
Open Virtualization Format (OVF and OVA)	OVF is an open standard for packaging and distributing a virtual appliance consisting of one or more VMs.
Disk image formats (VHD and VMDK)	Virtual Hard Disk (VHD) and Virtual Machine Disk (VMDK) format disk image files can be imported using the Import wizard. Importing a disk image may be appropriate when there is a virtual disk image available, with no OVF metadata associated.
XenServer XVA format	XVA is a format specific to Xen-based hypervisors for packaging an individual VM as a single file archive, including a descriptor and disk images. Its file name extension is `.xva`.

Which format to use?

Consider using OVF/OVA format to:

- Share XenServer vApps and VMs with other virtualization platforms that support OVF
- Save more than one VM
- Secure a vApp or VM from corruption and tampering
- Include a license agreement
- Simplify vApp distribution by storing an OVF package in an OVA file

Consider using XVA format to:

- Import and export VMs from a script with a CLI

Open virtualization format (OVF and OVA)

OVF is an open standard, specified by the Distributed Management Task Force, for packaging and distributing a virtual appliance consisting of one or more VMs. For further details about OVF and OVA formats, see the following information:

- Knowledge Base Article CTX121652: Overview of the Open Virtualization Format
- Open Virtualization Format Specification

> **Note:**
>
> To import OVF or OVA packages, you must be logged in as root or have the Pool Admin or Pool Operator RBAC role associated with your user account. To export a VM as an OVA or OVF package you must be logged in as root or have the Pool Admin, Pool Operator, VM Power Admin, or VM Admin RBAC role associated with your user account.

An **OVF Package** is the set of files that comprises the virtual appliance. It always includes a descriptor file and any other files that represent the following attributes of the package:

Attributes **Descriptor (`.ovf`):** The descriptor always specifies the virtual hardware requirements of the package. It may also specify other information, including:

- Descriptions of virtual disks, the package itself, and guest operating systems
- A license agreement
- Instructions to start and stop VMs in the appliance
- Instructions to install the package

Signature (`.cert`): The signature is the digital signature used by a public key certificate in the X.509 format to authenticate the author of the package.

Manifest (`.mf`): The manifest allows you to verify the integrity of the package contents. It contains the SHA-1 digests of every file in the package.

Virtual disks: OVF does not specify a disk image format. An OVF package includes files comprising virtual disks in the format defined by the virtualization product that exported the virtual disks. XenServer produces OVF packages with disk images in Dynamic VHD format; VMware products and Virtual Box produce OVF packages with virtual disks in Stream-Optimized VMDK format.

OVF packages also support other non-metadata related capabilities, such as compression, archiving, EULA attachment, and annotations.

> **Note:**
>
> When importing an OVF package that has been compressed or contains compressed files, you may need to free up extra disk space on the XenServer host to import it properly.

An **Open Virtual Appliance (OVA) package** is a single archive file, in the Tape Archive (.tar) format, containing the files that comprise an OVF Package.

Select OVF or OVA format OVF packages contain a series of uncompressed files, which makes it easier when you want to access individual disk images in the file. An OVA package contains one large file, and while you can compress this file, it does not give you the flexibility of a series of files.

Using the OVA format is useful for specific applications for which it is beneficial to have just one file, such as creating packages for Web downloads. Consider using OVA only as an option to make the package easier to handle. Using this format lengthens both the export and import processes.

Disk image formats (VHD and VMDK)

Using XenCenter, you can import disk images in the Virtual Hard Disk (VHD) and Virtual Machine Disk (VMDK) formats. Exporting standalone disk images is not supported.

> **Note:**
>
> To import disk images, ensure that you are logged in as root or have the Pool Administrator RBAC role associated with your user account.

You might choose to import a disk image when a virtual disk image is available without any associated OVF metadata. This option might occur in the following situations:

- It is possible to import a disk image, but the associated OVF metadata is not readable
- A virtual disk is not defined in an OVF package
- You are moving from a platform that does not allow you to create an OVF package (for example, older platforms or images)
- You want to import an older VMware appliance that does not have any OVF information
- You want to import a standalone VM that does not have any OVF information

When available, we recommend importing appliance packages that contain OVF metadata rather than an individual disk image. The OVF data provides information the Import wizard requires to recreate a VM from its disk image, This information includes the number of disk images associated with the VM, the processor, storage, network, memory requirements and so on. Without this information, it can be much more complex and error-prone to recreate the VM.

XVA format

XVA is a virtual appliance format specific to XenServer, which packages a single VM as a single set of files, including a descriptor and disk images. The file name extension is `.xva`.

The descriptor (file name extension `ova.xml`) specifies the virtual hardware of a single VM.

The disk image format is a directory of files. The directory name corresponds to a reference name in the descriptor and contains two files for each 1 MB block of the disk image. The base name of each file is the block number in decimal. The first file contains one block of the disk image in raw binary format and does not have an extension. The second file is a checksum of the first file. If the VM was exported from Citrix Hypervisor 8.0 or earlier, this file has the extension `.checksum`. If the VM was exported from Citrix Hypervisor 8.1 or later, this file has the extension `.xxhash`.

> **Important:**
>
> If a VM is exported from the XenServer host and then imported into another XenServer host with a different CPU type, it may not run properly. For example, a Windows VM exported from a host with an Intel® VT Enabled CPU might not run when imported into a host with an AMD-VTM CPU.

Operating system fixup

When importing a virtual appliance or disk image created and exported from a virtualization platform other than XenServer, you might have to configure the VM before it boots properly on the XenServer host.

XenCenter includes an advanced hypervisor interoperability feature –Operating System Fixup –which aims to ensure a basic level of interoperability for VMs that you import into XenServer. Use Operating System Fixup when importing VMs from OVF/OVA packages and disk images created on other virtualization platforms.

The Operating System Fixup process addresses the operating system device and driver issues inherent when moving from one hypervisor to another. The process attempts to repair boot device-related problems with the imported VM that might prevent the operating system within from booting in the XenServer environment. This feature is not designed to perform conversions from one platform to another.

> **Note:**
>
> This feature requires an ISO storage repository with 40 MB of free space and 256 MB of virtual memory.

Operating System Fixup is supplied as an automatically booting ISO image that is attached to the DVD drive of the imported VM. It performs the necessary repair operations when the VM is first started, and then shuts down the VM. The next time the new VM is started, the boot device is reset, and the VM starts normally.

To use Operating System Fixup on imported disk images or OVF/OVA packages, enable the feature on the Advanced Options page of the XenCenter Import wizard. Specify a location where the Fixup ISO is copied so that XenServer can use it.

What does operating system fixup do to the VM?

The Operating System Fixup option is designed to make the minimal changes possible to enable a virtual system to boot. Depending on the guest operating system and the hypervisor of the original host, further actions might be required after using Operating System Fixup. These actions can include configuration changes and driver installation.

During the Fixup process, an ISO is copied to an ISO SR. The ISO is attached to a VM. The boot order is set to boot from the virtual DVD drive, and the VM boots into the ISO. The environment within the ISO then checks each disk of the VM to determine if it is a Linux or a Windows system.

If a Linux system is detected, the location of the GRUB configuration file is determined. Any pointers to SCSI disk boot devices are modified to point to IDE disks. For example, if GRUB contains an entry of `/dev/sda1` representing the first disk on the first SCSI controller, this entry is changed to `/dev/hda1` representing the first disk on the first IDE controller.

If a Windows system is detected, a generic critical boot device driver is extracted from the driver database of the installed OS and registered with the OS. This process is especially important for older Windows operating systems when the boot device is changed between a SCSI and IDE interface.

If certain virtualization tool sets are discovered in the VM, they are disabled to prevent performance problems and unnecessary event messages.

Import VMs

When you import a VM, you effectively create a VM, using many of the same steps required to provision a new VM. These steps include nominating a host, and configuring storage and networking.

You can import OVF/OVA, disk image, XVA, and XVA Version 1 files using the XenCenter Import wizard. You can also import XVA files via the xe CLI.

Import VMs from OVF/OVA

> **Note:**
>
> To import OVF or OVA packages, you must be logged in as root or have the Pool Admin or Pool Operator RBAC role associated with your user account.

The XenCenter Import wizard allows you to import VMs that have been saved as OVF/OVA files. The Import wizard takes you through the usual steps to create a VM in XenCenter: nominating a host, and then configuring storage and networking for the new VM. When importing OVF and OVA files, extra steps may be required, such as:

- When importing VMs that have been created using other virtualization platforms, run the Operating System Fixup feature to ensure a basic level of interoperability for the VM. For more information, see Operating system fixup.

> **Tip:**
>
> Ensure that the target host has enough RAM to support the virtual machines being imported. A lack of available RAM results in a failed import. For more information about resolving this issue, see CTX125120 - Appliance Import Wizard Fails Because of Lack of Memory.

Imported OVF packages appear as vApps when imported using XenCenter. When the import is complete, the new VMs appear in the XenCenter **Resources** pane, and the new vApp appears in the **Manage vApps** dialog box.

To import VMs from OVF/OVA by using XenCenter:

1. Open the Import wizard by doing one of the following:

    - In the **Resources** pane, right-click, and then select **Import** on the shortcut menu.
    - On the **File** menu, select **Import**.

2. On the first page of the wizard, locate the file you want to import, and then click **Next** to continue.

3. Review and accept EULAs, if applicable.

    If the package you are importing includes any EULAs, accept them and click **Next** to continue. When no EULAs are included in the package, the wizard skips this step and advance straight to the next page.

4. Specify the pool or host to which you want to import the VMs, and then (optionally) assign the VMs to a home server.

    To select a host or pool, choose from the **Import VM(s)** to list.

    To assign each VM a home server, select a host from the list in the **Home Server**. If you want not to assign a home server, select **Don't assign a home server**.

    Click **Next** to continue.

5. Configure storage for the imported VMs: Choose one or more storage repositories on which to place the imported virtual disks, and then click **Next** to continue.

    To place all the imported virtual disks on the same SR, select **Place all imported VMs on this target SR**. Select an SR from the list.

    To place the virtual disks of incoming VMs onto different SRs, select **Place imported VMs on the specified target SRs**. For each VM, select the target SR from the list in the SR column.

6. Configure networking for the imported VMs: map the virtual network interfaces in the VMs you are importing to target networks in the destination pool. The Network and MAC address shown in the list of incoming VMs are stored as part of the definition of the original (exported) VM in the

export file. To map an incoming virtual network interface to a target network, select a network from the list in the Target Network column. Click **Next** to continue.

7. Specify security settings: If the selected OVF/OVA package is configured with security features, such as certificates or a manifest, specify the information necessary, and then click **Next** to continue.

   Different options appear on the Security page depending on which security features have been configured on the OVF appliance:

   - If the appliance is signed, a **Verify digital signature** check box appears, automatically selected. Click **View Certificate** to display the certificate used to sign the package. If the certificate appears as untrusted, it is likely that either the Root Certificate or the Issuing Certificate Authority is not trusted on the local computer. Clear the **Verify digital signature** check box if you do not want to verify the signature.
   - If the appliance includes a manifest, a **Verify manifest content** check box appears. Select this check box to have the wizard verify the list of files in the package.

   When packages are digitally signed, the associated manifest is verified automatically, so the **Verify manifest content** check box does not appear on the Security page.

   > **Note:**
   >
   > VMware Workstation 7.1.x OVF files fail to import when you choose to verify the manifest. This failure occurs because VMware Workstation 7.1.x produces an OVF file with a manifest that has invalid SHA-1 hashes. If you do not choose to verify the manifest, the import is successful.

8. Enable Operating System Fixup: If the VMs in the package you are importing were built on a virtualization platform other than XenServer, select the **Use Operating System Fixup** check box. Select an ISO SR where the Fixup ISO can be copied so that XenServer can access it. For more information about this feature, see Operating system fixup.

   Click **Next** to continue.

9. Review the import settings, and then click **Finish** to begin the import process and close the wizard.

   > **Note:**
   >
   > Importing a VM may take some time, depending on the size of the VM and the speed and bandwidth of the network connection.

The import progress is displayed in the status bar at the bottom of the XenCenter window and on the **Logs** tab. When the newly imported VM is available, it appears in the **Resources** pane, and the new vApp appears in the **Manage vApps** dialog box.

> **Note:**
>
> After using XenCenter to import an OVF package that contains Windows operating systems, you must set the `platform` parameter.
>
> 1. Set the `platform` parameter to `device_id=0002`. For example:
>
>    ```
>    xe vm-param-set uuid=VM uuid platform:device_id=0002
>    ```
>
> 2. Set the `platform` parameter to `viridian=true`. For example:
>
>    ```
>    xe vm-param-set uuid=VM uuid platform:viridian=true
>    ```

Import disk images

The XenCenter Import wizard allows you to import a disk image into a pool or specific host as a VM. The Import wizard takes you through the usual steps to create a VM in XenCenter: nominating a host, and then configuring storage and networking for the new VM.

Requirements

- You must be logged in as root or have the Pool Administrator Role Based Access Control (RBAC) role associated with your user account.
- Ensure that DHCP runs on the management network XenServer is using.
- The import wizard requires local storage on the server on which you are running it.

To import VMs from a Disk Image by using XenCenter:

1. Open the Import wizard by doing one of the following:
   - In the **Resources** pane, right-click, and then select **Import** on the shortcut menu.
   - On the **File** menu, select **Import**.
2. On the first page of the wizard, locate the file you want to import, and then click **Next** to continue.
3. Specify the VM name and allocate CPU and memory resources.

   Enter a name for the new VM to be created from the imported disk image, and then allocate the number of CPUs and amount of memory. Click **Next** to continue.
4. Specify the pool or host to which you want to import the VMs, and then (optionally) assign the VMs to a home server.

   To select a host or pool, choose from the **Import VM(s)** to list.

To assign each VM a home server, select a host from the list in the **Home Server**. If you want not to assign a home server, select **Don't assign a home server**.

Click **Next** to continue.

5. Configure storage for the imported VMs: Select one or more storage repositories on which to place the imported virtual disks, and then click **Next** to continue.

   To place all the imported virtual disks on the same SR, select **Place all imported VMs on this target SR**. Select an SR from the list.

   To place the virtual disks of incoming VMs onto different SRs, select **Place imported VMs on the specified target SRs**. For each VM, select the target SR from the list in the SR column.

6. Configure networking for the imported VMs: map the virtual network interfaces in the VMs you are importing to target networks in the destination pool. The Network and MAC address shown in the list of incoming VMs are stored as part of the definition of the original (exported) VM in the export file. To map an incoming virtual network interface to a target network, select a network from the list in the Target Network column. Click **Next** to continue.

7. Enable Operating System Fixup: If the disk images you are importing were built on a virtualization platform other than XenServer, select the Use Operating System Fixup check box. Select an ISO SR where the Fixup ISO can be copied so that XenServer can access it. For more information about this feature, see Operating system fixup.

   Click **Next** to continue.

8. Review the import settings, and then click **Finish** to begin the import process and close the wizard.

   > **Note:**
   >
   > Importing a VM may take some time, depending on the size of the VM and the speed and bandwidth of the network connection.

The import progress is displayed in the status bar at the bottom of the XenCenter window and on the **Logs** tab. When the newly imported VM is available, it appears in the **Resources** pane.

> **Note:**
>
> After using XenCenter to import a disk image that contains Windows operating systems, you must set the `platform` parameter. The value of this parameter varies according to the version of Windows contained in the disk image:

- For Windows Server 2016 and later, set the `platform` parameter to `device_id=0002`. For example:

```
xe vm-param-set uuid=VM uuid platform:device_id=0002
<!--NeedCopy-->
```

- For all other versions of Windows, set the `platform` parameter to `viridian=true`. For example:

  ```
  xe vm-param-set uuid=VM uuid platform:viridian=true
  <!--NeedCopy-->
  ```

Import VMs from XVA

You can import VMs, templates, and snapshots that have previously been exported and stored locally in XVA format (`.xva`). To do so, you follow the usual steps to create a VM: nominating a host, and then configuring storage and networking for the new VM.

> **Warning:**
>
> It may not always be possible to run an imported VM that was exported from another host with a different CPU type. For example, a Windows VM exported from a host with an Intel VT Enabled CPU might not run when imported to a host with an AMD-VTM CPU.

To import VMs from XVA by using XenCenter:

1. Open the Import wizard by doing one of the following:

   - In the **Resources** pane, right-click, and then select **Import** on the shortcut menu.
   - On the **File** menu, select **Import**.

2. On the first page of the wizard, locate the file you want to import (`.xva` or `ova.xml`), and then click **Next** to continue.

   If you enter a URL location (`http`, `https`, `file`, or `ftp`) in the **Filename** box. Click **Next**, a Download Package dialog box opens and you must specify a folder on your XenCenter host where the file is copied.

3. Select a pool or host for the imported VM to start on, and then choose **Next** to continue.

4. Select the storage repositories on which to place the imported virtual disk, and then click **Next** to continue.

5. Configure networking for the imported VM: map the virtual network interface in the VM you are importing to target a network in the destination pool. The Network and MAC address shown in the list of incoming VMs are stored as part of the definition of the original (exported) VM in the export file. To map an incoming virtual network interface to a target network, select a network from the list in the Target Network column. Click **Next** to continue.

6. Review the import settings, and then click **Finish** to begin the import process and close the wizard.

> **Note:**
>
> Importing a VM may take some time, depending on the size of the VM and the speed and bandwidth of the network connection.

The import progress is displayed in the status bar at the bottom of the XenCenter window and on the **Logs** tab. When the newly imported VM is available, it appears in the **Resources** pane.

To import a VM from XVA by using the xe CLI:

To import the VM to the default SR on the target XenServer host, enter the following:

```
xe vm-import -h hostname -u root -pw password \
    filename=pathname_of_export_file
<!--NeedCopy-->
```

To import the VM to a different SR on the target XenServer host, add the optional `sr-uuid` parameter:

```
xe vm-import -h hostname -u root -pw password \
    filename=pathname_of_export_file sr-uuid=uuid_of_target_sr
<!--NeedCopy-->
```

If you want to preserve the MAC address of the original VM, add the optional `preserve` parameter and set to `true`:

```
xe vm-import -h hostname -u root -pw password \
    filename=pathname_of_export_file preserve=true
<!--NeedCopy-->
```

> **Note:**
>
> Importing a VM may take some time, depending on the size of the VM and the speed and bandwidth of the network connection.

After the VM has been imported, the command prompt returns the UUID of the newly imported VM.

Export VMs

You can export OVF/OVA and XVA files using the XenCenter Export wizard; you can also export XVA files via the xe CLI.

Export VMs as OVF/OVA

Using the XenCenter Export wizard, you can export one or more VMs as an OVF/OVA package. When you export VMs as an OVF/OVA package, the configuration data is exported along with the virtual hard disks of each VM.

> **Note:**
>
> To export OVF or OVA packages, you must be logged in as root or have the Pool Admin, Pool Operator, VM Power Admin, or VM Admin RBAC role associated with your user account.

To export VMs as OVF/OVA by using XenCenter:

1. Shut down or suspend the VMs that you want to export.
2. Open the Export wizard: in the **Resources** pane, right-click the pool or host containing the VMs you want to export, and then select **Export**.
3. On the first page of the wizard:
   - Enter the name of the export file
   - Specify the folder where you want the files to be saved
   - Select **OVF/OVA Package (*.ovf, *.ova)** from the **Format** list
   - Click **Next** to continue
4. From the list of available VMs, select the VMs that you want to include in the OVF/OVA package, and then click **Next** to continue.
5. If necessary, you can add to a previously prepared End User Licensing Agreement (EULA) document (.rtf, .txt) to the package.

   To add a EULA, click **Add** and browse to the file you want to add. Once you have added the file, you can view the document by selecting it from the **EULA files** list and then clicking **View**.

   EULAs can provide the legal terms and conditions for using the appliance and the applications delivered in the appliance.

   The ability to include one or more EULAs lets you legally protect the software on the appliance. For example, if your appliance includes a proprietary operating system on its VMs, you might want to include the EULA text from that operating system. The text is displayed and the person who imports the appliance must accept it.

   > **Note:**
   >
   > Attempting to add EULA files that are not in supported formats, including XML or binary files, can cause the import EULA functionality to fail.

   Select **Next** to continue.
6. On the **Advanced options** page, specify a manifest, signature and output file options, or just click **Next** to continue.

   a) To create a manifest for the package, select the **Create a manifest** check box.

The manifest provides an inventory or list of the other files in a package. The manifest is used to ensure that the files originally included when the package was created are the same files present when the package arrives. When the files are imported, a checksum is used to verify that the files have not changed since the package was created.

b) To add a digital signature to the package

   i. Select **Sign the OVF package**.

      The digital signature (`.cert`) contains the signature of the manifest file and the certificate used to create that signature. When a signed package is imported, the user can verify the identity of the package creator by using the public key of the certificate to validate the digital signature.

   ii. Browse to locate a certificate.

      Use an X.509 certificate that you have already created from a Trusted Authority and exported as a `.pfx` file. For certificates with SHA-256 digest export using the "Microsoft Enhanced RSA and AES Cryptographic Provider"as CSP.

   iii. In **Private key password** enter the export (PFX) password, or, if an export password was not provided, the private key associated with the certificate.

c) To output the selected VMs as a single (tar) file in OVA format, select the **Create OVA package (single OVA export file)** check box. For more on the different file formats, see Open virtualization format.

d) To compress virtual hard disk images (.VHD files) included in the package, select the Compress OVF files check box.

   When you create an OVF package, the virtual hard disk images are, by default, allocated the same amount of space as the exported VM. For example, a VM that is allocated 26 GB of space has a hard disk image that consumes 26 GB of space. The hard disk image uses this space regardless of whether or not the VM actually requires it.

   > **Note:**
   >
   > Compressing the VHD files makes the export process take longer to complete. Importing a package containing compressed VHD files also takes longer, as the Import wizard must extract all of the VHD images as it imports them.

If both **Create OVA package (single OVA export file)** and **Compress OVF files** are checked, the result is a compressed OVA file with the extension `.ova.gz`.

7. Review the export settings.

   To have the wizard verify the exported package, select the **Verify export on completion** check box. Click **Finish** to begin the export process and close the wizard.

> **Note:**
>
> Exporting a VM may take some time, depending on the size of the VM and the speed and bandwidth of the network connection.

The export progress is displayed in the status bar at the bottom of the XenCenter window and on the **Logs** tab. To cancel an export in progress, click the **Logs** tab, find the export in the list of events, and click the **Cancel** button.

Export VMs as XVA You can export an existing VM as an XVA file using the XenCenter Export wizard or the xe CLI. We recommend exporting a VM to a machine other than the XenServer host, on which you can maintain a library of export files. For example, you can export the VM to the machine running XenCenter.

> **Warning:**
>
> It may not always be possible to run an imported VM that was exported from another host with a different CPU type. For example, a Windows VM exported from a host with an Intel VT Enabled CPU might not run when imported to a host with an AMD-VTM CPU.

To export VMs as XVA files by using XenCenter:

1. Shut down or suspend the VM that you want to export.
2. Open the Export wizard: from the **Resources** pane, right-click the VM which you want to export, and then select **Export**.
3. On the first page of the wizard:

    - Enter the name of the export file
    - Specify the folder where you want the files to be saved
    - Select **XVA File (*.xva)** from the **Format** list
    - Click **Next** to continue

4. From the list of available VMs, select the VM that you want to export, and then click **Next** to continue.
5. Review the export settings.

    To have the wizard verify the exported package, select the **Verify export on completion** check box. Click Finish to begin the export process and close the wizard.

    > **Note:**
    >
    > Exporting a VM may take some time, depending on the size of the VM and the speed and bandwidth of the network connection.

The export progress is displayed in the status bar at the bottom of the XenCenter window and on the **Logs** tab. To cancel an export in progress, click the **Logs** tab, find the export in the list of events, and click the **Cancel** button.

To export VMs as XVA files by using the xe CLI:

1. Shut down the VM that you want to export.

2. Export the VM by running the following:

   ```
   xe vm-export -h hostname -u root -pw password vm=vm_name \
       filename=pathname_of_file
   <!--NeedCopy-->
   ```

   > **Note:**
   >
   > Be sure to include the `.xva` extension when specifying the export file name. If the exported VM doesn't have this extension, XenCenter might fail to recognize the file as a valid XVA file when you attempt to import it.

Delete VMs

May 17, 2023

You can delete VMs by using the xe CLI or XenCenter.

Deleting a virtual machine (VM) removes its configuration and its filesystem from the host. When you delete a VM, you can choose to delete or preserve any virtual disks attached to the VM, in addition to any snapshots of the VM.

Delete a VM by using the xe CLI

To delete a VM:

1. Find the VM UUID:

   ```
   xe vm-list
   ```

2. Shutdown the VM:

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

3. (Optional) You can choose to delete the attached virtual disks:

   a) Find the virtual disk UUIDs:

```
xe vm-disk-list vm=<uuid>
```

b) Delete the virtual disk:

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

> **Important:**
>
> Any data stored in the VM's virtual disk drives is lost.

4. (Optional) You can choose to delete the snapshots associated with the VM:

   a) Find the UUIDs of the snapshots:

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

   b) For each snapshot to delete, find the UUIDs of the virtual disks for that snapshot:

   ```
   xe snapshot-disk-list snapshot-uuid=<uuid>
   ```

   c) Delete each snapshot disk:

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

   d) Delete the snapshot:

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

5. Delete the VM:

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

Delete a VM by using XenCenter

To delete a VM:

1. Shut down the VM.
2. Select the stopped VM in the **Resources** panel, right-click, and select **Delete** on the shortcut menu. Alternatively, on the **VM** menu, select **Delete**.
3. To delete an attached virtual disk, select its check box.

   > **Important:**
   >
   > Any data stored in the VM's virtual disk drives is lost.

4. To delete a snapshot of the VM, select its check box.

5. Click **Delete**.

   When the delete operation is completed, the VM is removed from the **Resources** pane.

> **Note:**
>
> VM snapshots whose parent VM has been deleted (*orphan snapshots*) can still be accessed from the **Resources** pane. These snapshots can be exported, deleted, or used to create VMs and templates. To view snapshots in the **Resources** pane, select **Objects** in the Navigation pane and then expand the **Snapshots** group in the Resources pane.

vApps

May 17, 2023

A vApp is a logical group of one or more related Virtual Machines (VMs) which can be started up as a single entity. When a vApp is started, the VMs contained within the vApp start in a user-predefined order. This feature enables VMs which depend upon one another to be automatically sequenced. An administrator no longer has to manually sequence the startup of dependent VMs when a whole service requires restarting (for instance for a software update). The VMs within the vApp do not have to reside on one host and can be distributed within a pool using the normal rules.

The vApp feature is useful in the Disaster Recovery situation. You can group all VMs that are on the same Storage Repository or all VMs that relate to the same Service Level Agreement (SLA).

> **Note:**
>
> vApps can be created and changed using both XenCenter and the xe CLI. For information on working with vApps using the CLI, see Command Line Interface.

Manage vApps in XenCenter

The **Manage vApps** dialog box enables you to create, delete, change, start, and shut down vApps, and import and export vApps within the selected pool. If you select a vApp in the list, the VMs it contains are listed in the details pane on the right.

You can use **Manage vApps** to do the following actions:

- To change the name or description of a vApp
- To add or remove VMs from the vApp
- To change the startup sequence of the VMs in the vApp

To change vApps:

1. Select the pool and, on the **Pool** menu, select **Manage vApps**.

   Alternatively, right-click in the **Resources** pane and select **Manage vApps** on the shortcut menu.

2. Select the vApp and choose **Properties** to open its Properties dialog box.
3. Select the **General** tab to change the vApp name or description.
4. Select the **Virtual Machines** tab to add or remove VMs from the vApp.
5. Select the **VM Startup Sequence** tab to change the start order and delay interval values for individual VMs in the vApp.
6. Click **OK** to save your changes and close **Properties**.

Create vApps

To group VMs together in a vApp follow the procedure:

1. Choose the pool and, on the **Pool** menu, select **Manage vApps**.
2. Type a name for the vApp, and optionally a description. Click **Next**.

   You can choose any name you like, but a name that describes the vApp is best. Although it is advisable to avoid creating multiple vApps that have the same name, it is not a requirement. XenCenter does not force vApp names to be unique. It is not necessary to use quotation marks for names that include spaces.

3. Choose which VMs to include in the new vApp. Click **Next**.

   You can use the search field to list only VMs that have names that include the specified text string.

4. Specify the startup sequence for the VMs in the vApp. Click **Next**.

Value	Description
Start Order	Specifies the order in which individual VMs are started up within the vApp, allowing certain VMs to be restarted before others. VMs that have a start order value of 0 (zero) are started first. VMs that have a start order value of 1 are started next. Then VMs that have a start order value of 2 are started, and so on.

Value	Description
Attempt to start next VM after	Specifies how long to wait after starting the VM before attempting to start the next group of VMs in the startup sequence. That next group is the set of VMs that have a lower start order.

1. On the final page of **Manage vApps**, you can review the vApp configuration. Click **Previous** to go back and change any settings or **Finish** to create the vApp and close **Manage vApps**.

    > **Note:**
    >
    > A vApp can span across multiple hosts in a single pool, but cannot span across several pools.

Delete vApps

To delete a vApp, follow the procedure:

1. Choose the pool and, on the **Pool** menu, select **Manage vApps**.
2. Select the vApp you want to delete from the list. Click **Delete**.

> **Note:**
>
> The VMs in the vApp are **not** deleted.

Start and shut down vApps by using XenCenter

To start or shut down a vApp, use **Manage vApps**, accessed from the **Pool** menu. When you start a vApp, all the VMs within it are started up automatically in sequence. The start order and delay interval values specified for each individual VM control the startup sequence. These values can be set when you first create the vApp. Change these values at any time from the vApp Properties dialog box or individual VM Properties dialog box.

To start a vApp:

1. Open **Manage vApps**: Choose the pool where the VMs in the vApp are located and, on the **Pool** menu, select **Manage vApps**. Alternatively, right-click in the **Resources** pane and select **Manage vApps** on the shortcut menu.
2. Choose the vApp and click **Start** to start all the VMs it contains.

To shut down a vApp:

1. Open **Manage vApps**: Choose the pool where the VMs in the vApp are located and, on the **Pool** menu, select **Manage vApps**. Alternatively, right-click in the **Resources** pane and select **Manage vApps** on the shortcut menu.
2. Choose the vApp and click **Shut Down** to shut down all the VMs in the vApp.

   A soft shutdown is attempted on all VMs. If a soft shutdown is not possible, then a forced shutdown is performed.

> **Note:**
>
> A soft shutdown performs a graceful shutdown of the VM, and all running processes are halted individually.
>
> A forced shutdown performs a hard shutdown and is the equivalent of unplugging a physical server. It might not always shut down all running processes. If you shut down a VM in this way, you risk losing data. Only use a forced shutdown when a soft shutdown is not possible.

Import and export vApps

vApps can be imported and exported as OVF/OVA packages. For more information, see Import and Export VMs.

To export a vApp:

1. Open **Manage vApps**: on the **Pool** menu, select **Manage vApps**.
2. Choose the vApp you want to export in the list. Click **Export**.
3. Follow the procedure described in Export VMs as OVF/OVA.

Exporting a vApp can take some time.

To import a vApp:

1. Open **Manage vApps**: on the **Pool** menu, select **Manage vApps**.
2. Click **Import** to open the **Import** dialog box.
3. Follow the procedure described in Import VMs as OVF/OVA.

After the import is complete, the new vApp appears in the list of vApps in **Manage vApps**.

Advanced notes for virtual machines

January 16, 2024

This section provides some advanced notes for Virtual Machines.

VM boot behavior

There are two options for the behavior of a Virtual Machine's VDI when the VM is booted:

> **Note:**
>
> The VM must be shut down before you can change its boot behavior setting.

Persist

> **Tip:**
>
> Use this boot behavior if you are hosting Citrix Virtual Desktops that are static or dedicated machines.

This behavior is the default on VM boot. The VDI is left in the state it was at the last shutdown.

Select this option if you plan to allow users to make permanent changes to their desktops. To select persist, shut down the VM, and then enter the following command:

```
xe vdi-param-set uuid=vdi_uuid on-boot=persist
<!--NeedCopy-->
```

Reset

> **Tip:**
>
> Use this boot behavior if you are hosting Citrix Virtual Desktops that are shared or randomly allocated machines.

On VM boot, the VDI is reverted to the state it was in at the previous boot. Any changes made while the VM is running are lost when the VM is next booted.

Select this option if you plan to deliver standardized desktops that users cannot permanently change. To select reset, shut down the VM, and then enter the following command:

> **Warning:**
>
> After you change `on-boot=reset`, any data saved to the VDI is discarded after the next shutdown/start or reboot.

Make the ISO library available to XenServer hosts

To make an ISO library available to XenServer hosts, create an external NFS or SMB/CIFS share directory. The NFS or SMB/CIFS server must allow root access to the share. For NFS shares, allow access by setting the no_root_squash flag when you create the share entry in /etc/exports on the NFS server.

Then either use XenCenter to attach the ISO library, or connect to the host console and run the command:

```
xe-mount-iso-sr host:/volume
<!--NeedCopy-->
```

For advanced use, you can pass extra arguments to the mount command.

To make a Windows SMB/CIFS share available to the host, either use XenCenter, or connect to the host console and run the following command:

```
xe-mount-iso-sr unc_path -t cifs -o username=myname/myworkgroup
<!--NeedCopy-->
```

Replace back slashes in the unc_path argument with forward-slashes. For example:

```
xe-mount-iso-sr //server1/myisos -t cifs -o username=johndoe/mydomain
<!--NeedCopy-->
```

After mounting the share, any available ISOs are available from the **Install from ISO Library or DVD drive** list in XenCenter. These ISOs are also available as CD images from the CLI commands.

Attach the ISO to an appropriate Windows template.

Connect to a Windows VM by using Remote Desktop

You can use one of the following ways of viewing a Windows VM console, both of which support full use of the keyboard and mouse.

- Using XenCenter. This method provides a standard graphical console and uses the VNC technology built in to XenServer to provide remote access to your virtual machine console.
- Connecting using Windows Remote Desktop. This method uses the Remote Desktop Protocol technology

In XenCenter on the **Console** tab, there is a **Switch to Remote Desktop** button. This button disables the standard graphical console within XenCenter, and switches to using Remote Desktop.

If you do not have Remote Desktop enabled in the VM, this button is disabled. To enable it, install the XenServer VM Tools for Windows. Follow the procedure below to enable it in each VM that you want to connect using Remote Desktop.

To enable Remote Desktop on a Windows VM:

1. Open **System** by clicking the **Start** button, right-click on **Computer**, and then select **Properties**.

2. Click **Remote settings**. If you're prompted for an administrator password, type the password you created during the VM setup.
3. In the **Remote Desktop** area, click the check box labeled **Allow connections from computers running any version of Remote Desktop**.
4. To select any non-administrator users that can connect to this Windows VM, click the **Select Remote Users** button and provide the user names. Users with Administrator privileges on the Windows domain can connect by default.

You can now connect to this VM using Remote Desktop. For more information, see the Microsoft Knowledge Base article, Connect to another computer using Remote Desktop Connection.

> **Note:**
>
> You cannot connect to a VM that is asleep or hibernating. Set the settings for sleep and hibernation on the remote computer to **Never**.

Time handling in Windows VMs

For Windows guests, initially the control domain clock drives the time. The time updates during VM lifecycle operations such as suspend and reboot.We recommend running a reliable NTP service in the control domain and all Windows VMs.

If you manually set a VM to be two hours ahead of the control domain, then it persists. You might set the VM ahead by using a time-zone offset within the VM. If you later change the control domain time (either manually or by NTP), the VM shifts accordingly but maintains the two hours offset. Changing the control domain time-zone does not affect VM time-zones or offset. XenServer uses the hardware clock setting of the VM to synchronize the VM. XenServer does not use the system clock setting of the VM.

When performing suspend and resume operations or using live migration, ensure that you have up-to-date XenServer VM Tools for Windows installed. XenServer VM Tools for Windows notify the Windows kernel that a time synchronization is required after resuming (potentially on a different physical host).

> **Note:**
>
> If you are running Windows VMs in Citrix Virtual Desktops environment, you must ensure that the host clock has the same source as the Active Directory (AD) domain. Failure to synchronize the clocks can cause the VMs to display an incorrect time and cause the Windows PV drivers to crash.

Time handling in Linux VMs

In addition to the behavior defined by XenServer, operating system settings and behaviors can affect the time handling behavior of your Linux VMs. Some Linux operating systems might periodically synchronize their system clock and hardware clock, or the operating system might use its own NTP service by default. For more information, see the documentation for the operating system of your Linux VM.

> **Note:**
>
> When installing a new Linux VM, ensure that you change the time-zone from the default UTC to your local value. For specific distribution instructions, see Linux Release Notes.

Hardware clocks in Linux VMs are **not** synchronized to the clock running on the control domain and can be altered. When the VM first starts, the control domain time is used to set the initial time of the hardware clock and system clock.

If you change the time on the hardware clock, this change is persisted when the VM reboots.

System clock behavior depends on the operating system of the VM. For more information, see the documentation for your VM operating system.

You cannot change this XenServer time handling behavior.

Install VMs from Reseller Option Kit (BIOS-locked) media

There are two types of VM: BIOS-generic and BIOS-customized. To enable installation of Reseller Option Kit (BIOS-locked) OEM versions of Windows onto a VM, copy the BIOS strings of the VM from the host with which the media was supplied. Alternatively, advanced users can set user-defined values to the BIOS strings.

BIOS-generic

The VM has generic XenServer BIOS strings.

> **Note:**
>
> If a VM doesn't have BIOS strings set when it starts, the standard XenServer BIOS strings are inserted into it and the VM becomes BIOS-generic.

BIOS-customized

You can customize the BIOS in two ways: Copy-Host BIOS strings and User-Defined BIOS strings.

> **Note:**
>
> After you first start a VM, you cannot change its BIOS strings. Ensure that the BIOS strings are correct before starting the VM for the first time.

Copy-Host BIOS strings The VM has a copy of the BIOS strings of a particular host in the pool. To install the BIOS-locked media that came with your host, follow the procedures given below.

Using XenCenter:

1. Click the **Copy host BIOS strings to VM** check box in the New VM Wizard.

Using the CLI:

1. Run the `vm-install copy-bios-strings-from` command. Specify the `host-uuid` as the host from which the strings are copied (that is, the host that the media was supplied with):

   ```
   xe vm-install copy-bios-strings-from=host uuid \
       template=template name sr-name-label=name of sr \
       new-name-label=name for new VM
   <!--NeedCopy-->
   ```

   This command returns the UUID of the newly created VM.

   For example:

   ```
   xe vm-install copy-bios-strings-from=46dd2d13-5aee-40b8-ae2c-95786
      ef4 \
       template="win7sp1" sr-name-label=Local\ storage  \
       new-name-label=newcentos
       7cd98710-bf56-2045-48b7-e4ae219799db
   <!--NeedCopy-->
   ```

2. If the relevant BIOS strings from the host have been successfully copied into the VM, the command `vm-is-bios-customized` confirms this success:

   ```
   xe vm-is-bios-customized uuid=VM uuid
   <!--NeedCopy-->
   ```

   For example:

   ```
   xe vm-is-bios-customized uuid=7cd98710-bf56-2045-48b7-e4ae219799db
       This VM is BIOS-customized.
   <!--NeedCopy-->
   ```

   > **Note:**
   >
   > When you start the VM, it is started on the physical host from which you copied the BIOS

> strings.

> **Warning:**
>
> It is your responsibility to comply with any EULAs governing the use of any BIOS-locked operating systems that you install.

User-defined BIOS strings The user has option to set custom values in selected BIOS strings using CLI/API. To install the media in a VM with customized BIOS, follow the procedure given below.

Using the CLI:

1. Run the `vm-install` command (without `copy-bios-strings-from`):

   ```
   xe vm-install template=template name sr-name-label=name of sr \
         new-name-label=name for new VM
   <!--NeedCopy-->
   ```

   This command returns the UUID of the newly created VM.

   For example:

   ```
   xe vm-install template="win7sp1" sr-name-label=Local\ storage  \
         new-name-label=newcentos
       7cd98710-bf56-2045-48b7-e4ae219799db
   <!--NeedCopy-->
   ```

2. To set user-defined BIOS strings, run the following command before starting the VM for the first time:

   ```
   xe vm-param-set uuid=VM_UUID bios-strings:bios-vendor=VALUE \
       bios-strings:bios-version=VALUE bios-strings:system-
          manufacturer=VALUE \
       bios-strings:system-product-name=VALUE bios-strings:system-
          version=VALUE \
       bios-strings:system-serial-number=VALUE bios-strings:enclosure
          -asset-tag=VALUE
   <!--NeedCopy-->
   ```

   For example:

   ```
   xe vm-param-set uuid=7cd98710-bf56-2045-48b7-e4ae219799db \
       bios-strings:bios-vendor="vendor name" \
       bios-strings:bios-version=2.4 \
       bios-strings:system-manufacturer="manufacturer name" \
       bios-strings:system-product-name=guest1 \
       bios-strings:system-version=1.0 \
       bios-strings:system-serial-number="serial number" \
       bios-strings:enclosure-asset-tag=abk58hr
   <!--NeedCopy-->
   ```

> **Notes:**
>
> - Once the user-defined BIOS strings are set in a single CLI/API call, they cannot be modified.
> - You can decide on the number of parameters you want to provide to set the user-defined BIOS strings.

> **Warning:**
>
> It is your responsibility to:
>
> - Comply with any EULAs and standards for the values being set in VM's BIOS.
> - Ensure that the values you provide for the parameters are working parameters. Providing incorrect parameters can lead to boot/media installation failure.

Assign a GPU to a Windows VM (for use with Citrix Virtual Desktops)

XenServer enables you to assign a physical GPU in the XenServer host to a Windows VM running on the same host. This GPU pass-through feature benefits graphics power users, such as CAD designers, who require high performance graphics capabilities. It is supported only for use with Citrix Virtual Desktops.

While XenServer supports only one GPU for each VM, it automatically detects and groups identical physical GPUs across hosts in the same pool. Once assigned to a group of GPUs, a VM may be started on any host in the pool that has an available GPU in the group. When attached to a GPU, a VM has certain features that are no longer available, including live migration, VM snapshots with memory, and suspend/resume.

Assigning a GPU to a VM in a pool does not interfere with the operation of other VMs in the pool. However, VMs with GPUs attached are considered non-agile. If VMs with GPUs attached are members of a pool with high availability enabled, both features overlook these VMs. The VMs cannot be migrated automatically.

GPU pass-through can be enabled using XenCenter or the xe CLI.

Requirements

GPU pass-through is supported for specific machines and GPUs. In all cases, the IOMMU chipset feature (known as VT-d for Intel models) must be available and enabled on the XenServer host. Before enabling the GPU pass-through feature, visit the Hardware Compatibility List.

Before assigning a GPU to a VM

Before you assign a GPU to a VM, put the appropriate physical GPUs in your XenServer host and then restart the machine. Upon restart, XenServer automatically detects any physical GPUs. To view all physical GPUs across hosts in the pool, use the `xe pgpu-list` command.

Ensure that the IOMMU chipset feature is enabled on the host. To do so, enter the following:

```
xe host-param-get uuid=uuid_of_host param-name=chipset-info param-key=
    iommu
<!--NeedCopy-->
```

If the value printed is **`false`**, IOMMU is not enabled, and GPU pass-through is not available using the specified XenServer host.

To assign a GPU to a Windows VM by using XenCenter:

1. Shut down the VM that you want to assign a GPU.
2. Open the VM properties: right-click the VM and select **Properties**.
3. Assign a GPU to the VM: Select GPU from the list of VM properties, and then select a GPU type. Click **OK**.
4. Start the VM.

To assign a GPU to a Windows VM by using the xe CLI:

1. Shut down the VM that you want to assign a GPU group by using the `xe vm-shutdown` command.
2. Find the UUID of the GPU group by entering the following:

   ```
   xe gpu-group-list
   <!--NeedCopy-->
   ```

   This command prints all GPU groups in the pool. Note the UUID of the appropriate GPU group.
3. Attach the VM to a GPU group by entering the following:

   ```
   xe vpgu-create gpu-group-uuid=uuid_of_gpu_group vm-uuid=uuid_of_vm
   <!--NeedCopy-->
   ```

   To ensure that the GPU group has been attached, run the `xe vgpu-list` command.
4. Start the VM by using the `xe vm-start` command.
5. Once the VM starts, install the graphics card drivers on the VM.

   Installing the drivers is essential, as the VM has direct access to the hardware on the host. Drivers are provided by your hardware vendor.

> **Note:**
>
> If you try to start a VM with GPU pass-through on the host without an available GPU in the appropriate GPU group, XenServer prints an error.

To detach a Windows VM from a GPU by using XenCenter:

1. Shut down the VM.
2. Open the VM properties: right-click the VM and select **Properties**.
3. Detach the GPU from the VM: Select **GPU** from the list of VM properties, and then select **None** as the GPU type. Click **OK**.
4. Start the VM.

To detach a Windows VM from a GPU by using the xe CLI:

1. Shut down the VM by using the `xe vm-shutdown` command.
2. Find the UUID of the vGPU attached to the VM by entering the following:

   ```
   xe vgpu-list vm-uuid=uuid_of_vm
   <!--NeedCopy-->
   ```

3. Detach the GPU from the VM by entering the following:

   ```
   xe vgpu-destroy uuid=uuid_of_vgpu
   <!--NeedCopy-->
   ```

4. Start the VM by using the `xe vm-start` command.

Create ISO images

XenServer can use ISO images as installation media and data sources for Windows or Linux VMs. This section describes how to make ISO images from CD/DVD media.

To create an ISO on a Linux system:

1. Put the CD- or DVD-ROM disk into the drive. Ensure that the disk is not mounted. To check, run the command:

   ```
   mount
   <!--NeedCopy-->
   ```

   If the disk is mounted, unmount the disk. See your operating system documentation for assistance if necessary.

2. As root, run the command

```
dd if=/dev/cdrom of=/path/cdimg_filename.iso
<!--NeedCopy-->
```

This command takes some time. When the operation is completed successfully, you see something like:

```
1187972+0 records in
1187972+0 records out
<!--NeedCopy-->
```

Your ISO file is ready.

To create an ISO on a Windows system:

Windows computers do not have an equivalent operating system command to create an ISO. Most CD-burning tools have a means of saving a CD as an ISO file.

Enable VNC for Linux VMs

January 18, 2024

VMs might not be set up to support Virtual Network Computing (VNC), which XenServer uses to control VMs remotely, by default. Before you can connect with XenCenter, ensure that the VNC server and an X display manager are installed on the VM and properly configured. This section describes how to configure VNC on each of the supported Linux operating system distributions to allow proper interactions with XenCenter.

For CentOS-based VMs, use the instructions for the Red Hat-based VMs below, as they use the same base code to provide graphical VNC access. CentOS *X* is based on Red Hat Enterprise Linux *X*.

Enable a graphical console on Debian VMs

> **Note:**
>
> Before enabling a graphical console on your Debian VM, ensure that you have installed the XenServer VM Tools for Linux. For more information, see Install the XenServer VM Tools for Linux.

The graphical console for Debian virtual machines is provided by a VNC server running inside the VM. In the recommended configuration, a standard display manager controls the console so that a login dialog box is provided.

1. Install your Debian guest with the desktop system packages, or install GDM (the display manager) using apt (following standard procedures).

2. Install the Xvnc server using `apt-get` (or similar):

```
apt-get install vnc4server
<!--NeedCopy-->
```

> **Note:**
>
> The Debian Graphical Desktop Environment, which uses the Gnome Display Manager version 3 daemon, can take significant CPU time. Uninstall the Gnome Display Manager `gdm3` package and install the `gdm` package as follows:

```
apt-get install gdm
apt-get purge gdm3
<!--NeedCopy-->
```

3. Set up a VNC password (not having one is a serious security risk) by using the `vncpasswd` command. Pass in a file name to write the password information to. For example:

```
vncpasswd /etc/vncpass
<!--NeedCopy-->
```

4. Modify your `gdm.conf` file (`/etc/gdm/gdm.conf`) to configure a VNC server to manage display `0` by extending the `[servers]` and `[daemon]` sections as follows:

```
    [servers]
    0=VNC
    [daemon]
    VTAllocation=false
    [server-VNC]
    name=VNC
    command=/usr/bin/Xvnc -geometry 800x600 -PasswordFile /etc/
        vncpass BlacklistTimeout=0
    flexible=true
<!--NeedCopy-->
```

5. Restart GDM, and then wait for XenCenter to detect the graphical console:

```
/etc/init.d/gdm restart
<!--NeedCopy-->
```

> **Note:**
>
> You can check that the VNC server is running using a command like `ps ax | grep vnc`.

Enable a graphical console on Red Hat, CentOS, or Oracle Linux VMs

> **Note:**
>
> Before setting up your Red Hat VMs for VNC, be sure that you have installed the XenServer VM

> Tools for Linux. For more information, see Install the XenServer VM Tools for Linux.

To configure VNC on Red Hat VMs, modify the GDM configuration. The GDM configuration is held in a file whose location varies depending on the version of Red Hat Linux you are using. Before modifying it, first determine the location of this configuration file. This file is modified in several subsequent procedures in this section.

Determine the location of your VNC configuration file

If you are using Red Hat Linux, the GDM configuration file is `/etc/gdm/custom.conf`. This file is a split configuration file that contains only user-specified values that override the default configuration. This type of file is used by default in newer versions of GDM. It is included in these versions of Red Hat Linux.

Configure GDM to use VNC

1. As root on the text CLI in the VM, run the command `rpm -q vnc-server gdm`. The package names `vnc-server` and `gdm` appear, with their version numbers specified.

   The package names that are displayed show the packages that are already installed. If you see a message that says that a package is not installed, you might have not selected the graphical desktop options during installation. Install these packages before you can continue. For details regarding installing more software on your VM, see the appropriate Red Hat Linux x86 Installation Guide.

2. Open the GDM configuration file with your preferred text editor and add the following lines to the file:

   ```
       [server-VNC]
       name=VNC Server
       command=/usr/bin/Xvnc -SecurityTypes None -geometry 1024x768 -
          depth 16 \
       -BlacklistTimeout 0
       flexible=true
   <!--NeedCopy-->
   ```

   With configuration files on Red Hat Linux, add these lines into the empty `[servers]` section.

3. Modify the configuration so that the `Xvnc` server is used instead of the standard X server:

   - `0=Standard`

     Modify it to read:

     `0=VNC`

    - If you are using Red Hat Linux, add the above line just below the `[servers]` section and before the `[server-VNC]` section.

4. Save and close the file.

Restart GDM for your change in configuration to take effect, by running the command `/usr/sbin/gdm-restart`.

> **Note:**
>
> Red Hat Linux uses runlevel 5 for graphical startup. If your installation starts up in runlevel 3, change this configuration for the display manager to be started and get access to a graphical console. For more information, see Check Run levels.

Firewall settings

The firewall configuration by default does not allow VNC traffic to go through. If you have a firewall between the VM and XenCenter, allow traffic over the port that the VNC connection uses. By default, a VNC server listens for connections from a VNC viewer on TCP port `5900 + n`, where `n` is the display number (usually zero). So a VNC server setup for Display-0 listens on TCP port `5900`, Display-1 is `TCP-5901`, and so on. Consult your firewall documentation to ensure that these ports are open. For more information, see Communication Ports Used by XenServer.

If you want to use IP connection tracking or limit the initiation of connections to be from one side only, further configure your firewall.

To configure Red Hat-base VMS firewall to open the VNC port:

1. For Red Hat Linux, use `system-config-securitylevel-tui`.
2. Select **Customize** and add `5900` to the other ports list.

Alternatively, you can disable the firewall until the next reboot by running the command `service iptables stop`, or permanently by running `chkconfig iptables off`. This configuration can expose extra services to the outside world and reduce the overall security of your VM.

VNC screen resolution

After connecting to a VM with the graphical console, the screen resolution sometimes doesn't match. For example, the VM display is too large to fit comfortably in the Graphical Console pane. Control this behavior by setting the VNC server `geometry` parameter as follows:

1. Open the GDM configuration file with your preferred text editor. For more information, see Determine the Location of your VNC Configuration File.

2. Find the `[server-VNC]` section you added above.
3. Edit the command line to read, for example:

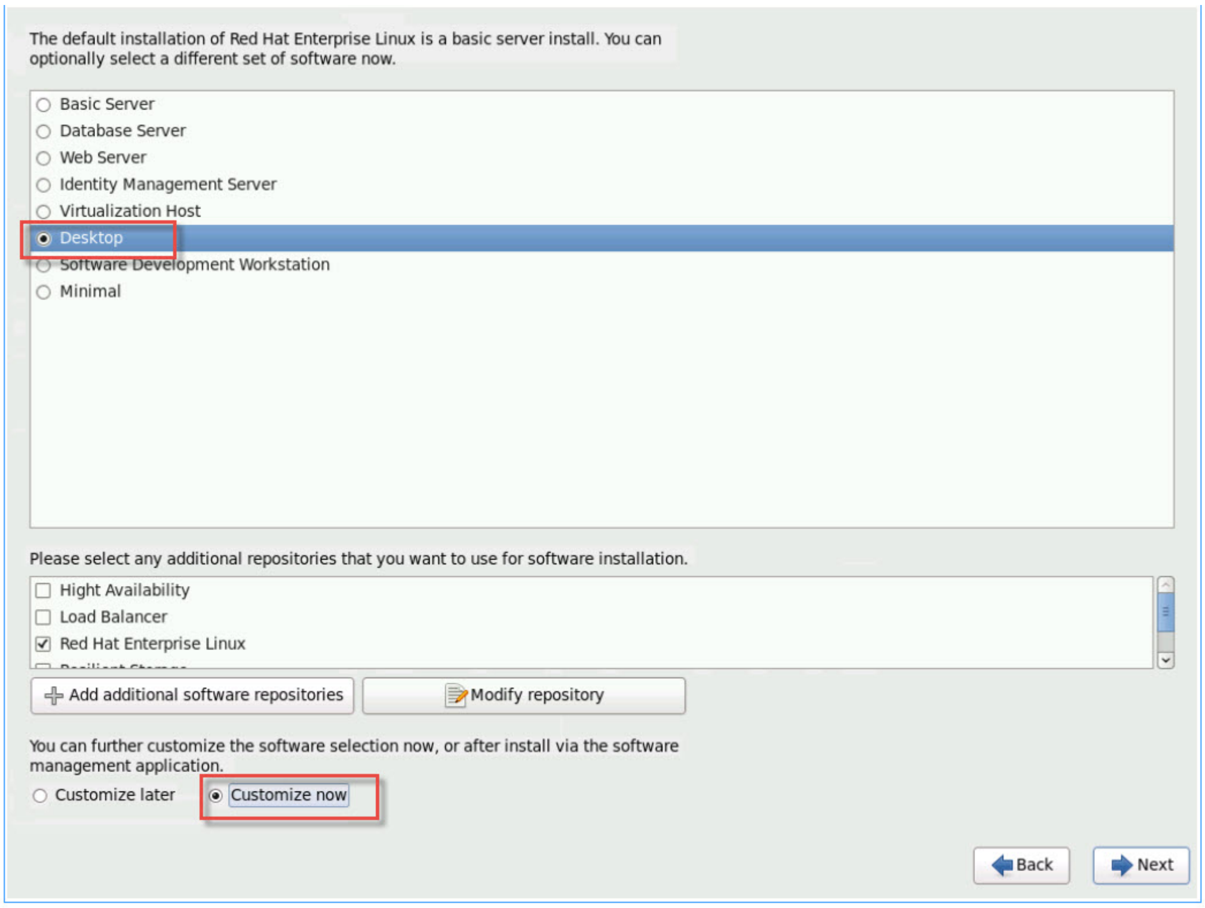
   ```
   command=/usr/bin/Xvnc -SecurityTypes None -geometry 800x600
   <!--NeedCopy-->
   ```

   The value of the `geometry` parameter can be any valid screen width and height.
4. Save and close the file.

Enable VNC for RHEL, CentOS, or OEL VMs

If you are using Red Hat Linux, the GDM configuration file is `/etc/gdm/custom.conf`. This file is a split configuration file that contains only user-specified values that override the default configuration. By default, this type of file is used in newer versions of GDM and is included in these versions of Red Hat Linux.

During the operating system installation, select **Desktop** mode. On the RHEL installation screen, select **Desktop** > **Customize now** and then click **Next**:



The default installation of Red Hat Enterprise Linux is a basic server install. You can optionally select a different set of software now.

- ○ Basic Server
- ○ Database Server
- ○ Web Server
- ○ Identity Management Server
- ○ Virtualization Host
- ● Desktop
- ○ Software Development Workstation
- ○ Minimal

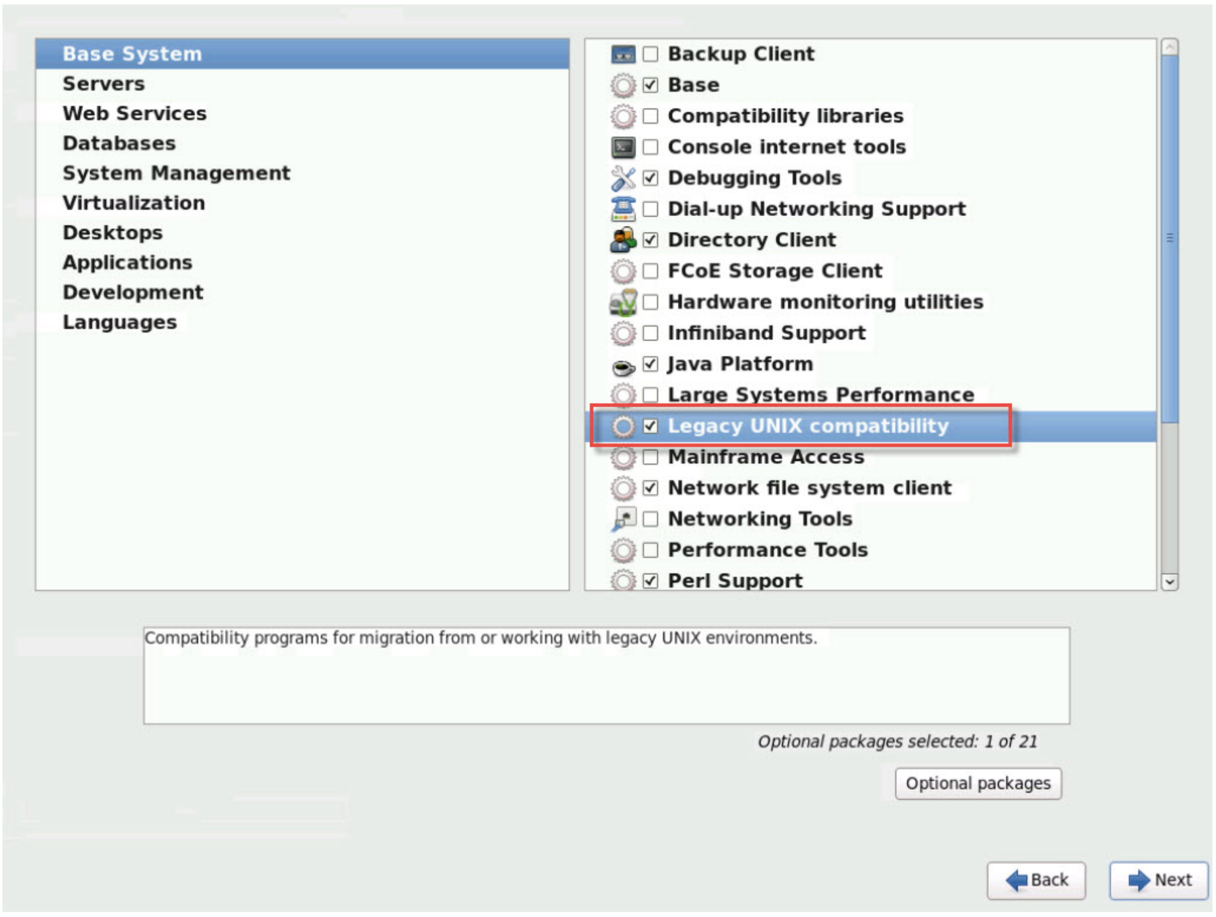
Please select any additional repositories that you want to use for software installation.

- ☐ Hight Availability
- ☐ Load Balancer
- ☑ Red Hat Enterprise Linux

You can further customize the software selection now, or after install via the software management application.

○ Customize later [✓ Customize now]

This action displays the Base System screen, ensure that **Legacy UNIX compatibility** is selected:

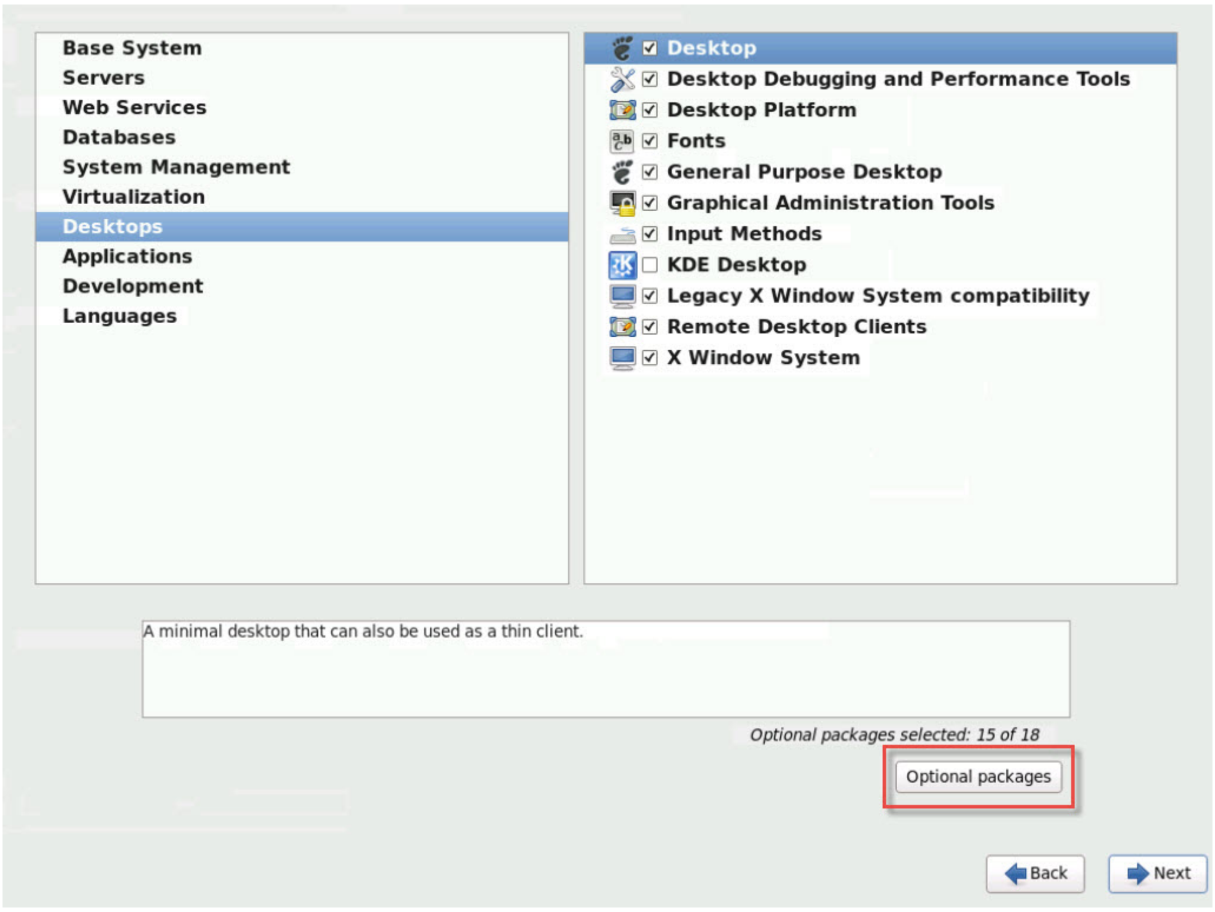


- [ ] Backup Client
- [x] Base
- [ ] Compatibility libraries
- [ ] Console internet tools
- [x] Debugging Tools
- [ ] Dial-up Networking Support
- [x] Directory Client
- [ ] FCoE Storage Client
- [ ] Hardware monitoring utilities
- [ ] Infiniband Support
- [x] Java Platform
- [ ] Large Systems Performance
- [x] Legacy UNIX compatibility
- [ ] Mainframe Access
- [x] Network file system client
- [ ] Networking Tools
- [ ] Performance Tools
- [x] Perl Support

Compatibility programs for migration from or working with legacy UNIX environments.

Optional packages selected: 1 of 21

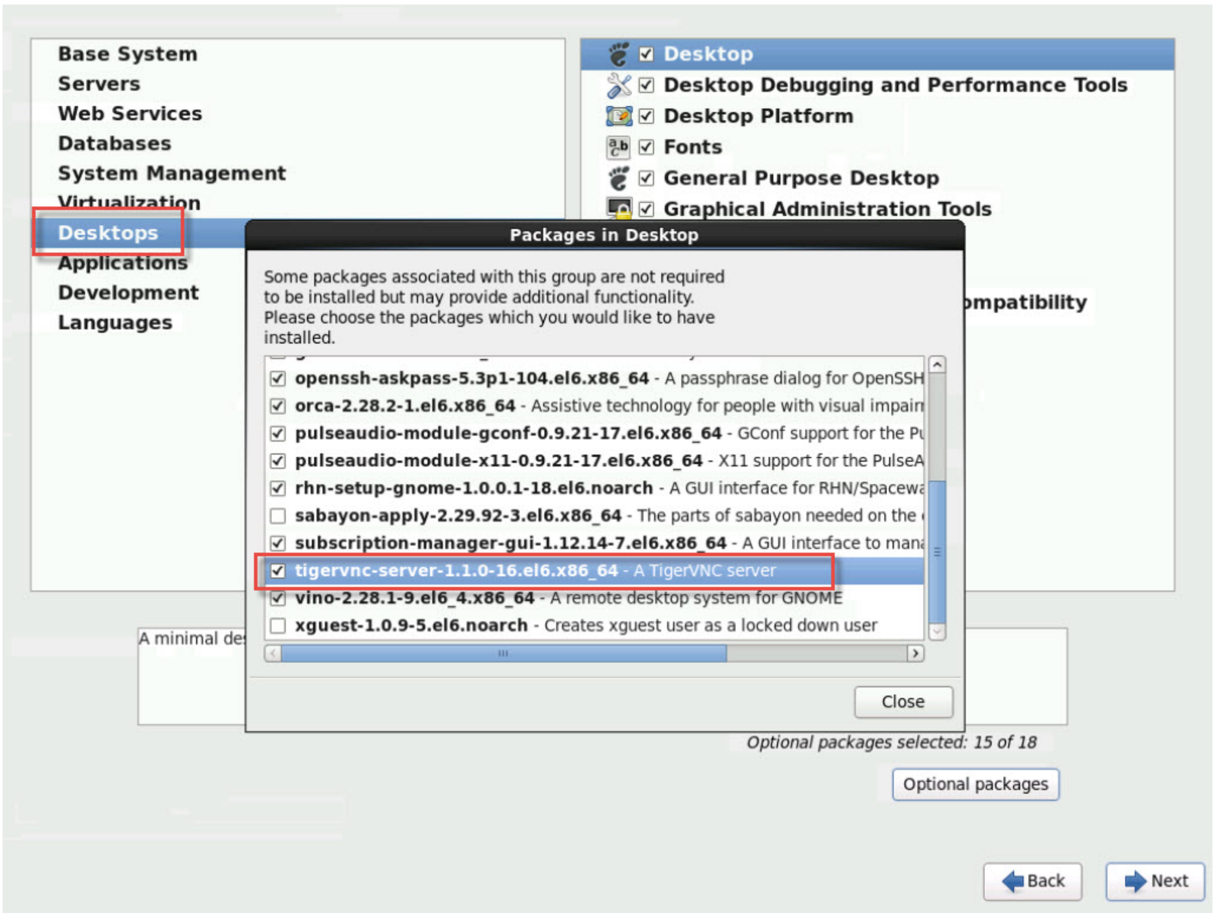
Select **Desktops** > **Optional packages**, then click **Next**:



A minimal desktop that can also be used as a thin client.

Optional packages selected: 15 of 18

This action displays the **Packages in Desktop** window, select **tigervnc-server-<version_number>** and then click **Next**:



☑ Desktop
☑ Desktop Debugging and Performance Tools
☑ Desktop Platform
☑ Fonts
☑ General Purpose Desktop
☑ Graphical Administration Tools

Some packages associated with this group are not required to be installed but may provide additional functionality. Please choose the packages which you would like to have installed.

☑ **openssh-askpass-5.3p1-104.el6.x86_64** - A passphrase dialog for OpenSSH
☑ **orca-2.28.2-1.el6.x86_64** - Assistive technology for people with visual impairn
☑ **pulseaudio-module-gconf-0.9.21-17.el6.x86_64** - GConf support for the Pu
☑ **pulseaudio-module-x11-0.9.21-17.el6.x86_64** - X11 support for the PulseA
☑ **rhn-setup-gnome-1.0.0.1-18.el6.noarch** - A GUI interface for RHN/Spacewa
☐ **sabayon-apply-2.29.92-3.el6.x86_64** - The parts of sabayon needed on the
☑ **subscription-manager-gui-1.12.14-7.el6.x86_64** - A GUI interface to mana
☑ **tigervnc-server-1.1.0-16.el6.x86_64** - A TigerVNC server
☑ **vino-2.28.1-9.el6_4.x86_64** - A remote desktop system for GNOME
☐ **xguest-1.0.9-5.el6.noarch** - Creates xguest user as a locked down user

A minimal de

Optional packages selected: 15 of 18

Work through the following steps to continue the setup of your RHEL VMs:

1. Open the GDM configuration file with your preferred text editor and add the following lines to the appropriate sections:

```
     [security]
     DisallowTCP=false

     [xdmcp]
     Enable=true
<!--NeedCopy-->
```

2. Create the file, `/etc/xinetd.d/vnc-server-stream`:

```
     service vnc-server
     {

                  id = vnc-server
             disable = no
                type = UNLISTED
                port = 5900
         socket_type = stream
                wait = no
                user = nobody
                group = tty
               server = /usr/bin/Xvnc
          server_args = -inetd -once -query localhost -
              SecurityTypes None \
          -geometry 800x600 -depth 16
      }

<!--NeedCopy-->
```

3. Enter the following command to start the `xinetd` service:

```
# service xinetd start
<!--NeedCopy-->
```

4. Open the file `/etc/sysconfig/iptables`. Add the following line above the line reading, `-A INPUT -j REJECT --reject-with icmp-host-prohibited`:

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5900 -j ACCEPT
<!--NeedCopy-->
```

5. Enter the following command to restart `iptables`:

```
# service iptables restart
<!--NeedCopy-->
```

6. Enter the following command to restart `gdm`:

```
    # telinit 3
    # telinit 5
<!--NeedCopy-->
```

> **Note:**
>
> Red Hat Linux uses runlevel 5 for graphical startup. If your installation starts up in runlevel 3, change this configuration for the display manager be started and to get access to a graphical console. For more information, see Check run levels.

Set up SLES-based VMs for VNC

> **Note:**
>
> Before setting up your SUSE Linux Enterprise Server VMs for VNC, be sure that you have installed the XenServer VM Tools for Linux. See Install the XenServer VM Tools for Linux for details.

SLES has support for enabling "Remote Administration"as a configuration option in `YaST`. You can select to enable Remote Administration at install time, available on the **Network Services** screen of the SLES installer. This feature allows you to connect an external VNC viewer to your guest to allow you to view the graphical console. The method for using the SLES remote administration feature is slightly different than the method provided by XenCenter. However, it is possible to modify the configuration files in your SUSE Linux VM such that it is integrated with the graphical console feature.

Check for a VNC server

Before making configuration changes, verify that you have a VNC server installed. SUSE ships the `tightvnc` server by default. This server is a suitable VNC server, but you can also use the standard RealVNC distribution.

You can check that you have the `tightvnc` software installed by running the command:

```
rpm -q tightvnc
<!--NeedCopy-->
```

Enable remote administration

If Remote Administration was not enabled during installation of the SLES software, you can enable it as follows:

1. Open a text console on the VM and run the `YaST` utility:

   ```
   yast
   <!--NeedCopy-->
   ```

2. Use the arrow keys to select **Network Services** in the left menu. **Tab** to the right menu and use the arrow keys to select **Remote Administration**. Press **Enter**.
3. In the **Remote Administration** screen, **Tab** to the **Remote Administration Settings** section. Use the arrow keys to select **Allow Remote Administration** and press **Enter** to place an X in the check box.
4. **Tab** to the **Firewall Settings** section. Use the arrow keys to select **Open Port in Firewall** and press **Enter** to place an X in the check box.
5. **Tab** to the **Finish** button and press **Enter**.
6. A message box is displayed, telling you to restart the display manager for your settings to take effect. Press **Enter** to acknowledge the message.
7. The original top-level menu of `YaST` appears. **Tab** to the **Quit** button and press **Enter**.

Modify the `xinetd` configuration

After enabling Remote Administration, modify a configuration file if you want to allow XenCenter to connect. Alternatively, use a third party VNC client.

1. Open the file `/etc/xinetd.d/vnc` in your preferred text editor.
2. The file contains sections like the following:

```
    service vnc1
    {

    socket_type = stream
    protocol    = tcp
    wait        = no
    user        = nobody
    server      = /usr/X11R6/bin/Xvnc
    server_args = :42 -inetd -once -query localhost -geometry 1024
       x768 -depth 16
    type        = UNLISTED
    port        = 5901
     }

<!--NeedCopy-->
```

3. Edit the `port` line to read

```
port = 5900
<!--NeedCopy-->
```

4. Save and close the file.
5. Restart the display manager and `xinetd` service with the following commands:

```
/etc/init.d/xinetd restart
rcxdm restart
<!--NeedCopy-->
```

SUSE Linux uses runlevel 5 for graphical startup. If your remote desktop does not appear, verify that your VM is configured to start up in runlevel 5. For more information, see Check Run levels.

Firewall settings

By default the firewall configuration does not allow VNC traffic to go through. If you have a firewall between the VM and XenCenter, allow traffic over the port that the VNC connection uses. By default, a VNC server listens for connections from a VNC viewer on TCP port `5900 + n`, where `n` is the display number (usually zero). So a VNC server setup for Display-0 listens on TCP port `5900`, Display-1 is `TCP -5901`, and so forth. Consult your firewall documentation to ensure that these ports are open. For more information, see Communication Ports Used by XenServer.

If you want to use IP connection tracking or limit the initiation of connections to be from one side only, further configure your firewall.

To Open the VNC Port on SLES 11.x VMs Firewall:

1. Open a text console on the VM and run the `YaST` utility:

   ```
   yast
   <!--NeedCopy-->
   ```

2. Use the arrow keys to select **Security and Users** in the left menu. **Tab** to the right menu and use the arrow keys to select **Firewall**. Press **Enter**.
3. In the **Firewall** screen, use the arrow keys to select **Custom Rules** in the left menu and then press **Enter**.
4. **Tab** to the **Add** button in the **Custom Allowed Rules** section and then press **Enter**.
5. In the **Source Network** field, enter *0/0*. **Tab** to the **Destination Port** field and enter *5900*.
6. **Tab** to the **Add** button and then press **Enter**.
7. **Tab** to the **Next** button and press **Enter**.
8. In the **Summary** screen **Tab** to the **Finish** button and press **Enter**.
9. On the top-level `YaST` screen **Tab** to the **Quit** button and press **Enter**.
10. Restart the display manager and `xinetd` service with the following commands:

    ```
    /etc/init.d/xinetd restart
    rcxdm restart
    <!--NeedCopy-->
    ```

Alternatively, you can disable the firewall until the next reboot by running the **rcSuSEfirewall2 stop** command, or permanently by using `YaST`. This configuration can expose extra services to the outside world and reduce the overall security of your VM.

VNC screen resolution

After connecting to a Virtual Machine with the Graphical Console, the screen resolution sometimes does not match. For example, the VM display is too large to fit comfortably in the Graphical Console pane. Control this behavior by setting the VNC server `geometry` parameter as follows:

1. Open the `/etc/xinetd.d/vnc` file with your preferred text editor and find the `service_vnc1` section (corresponding to `displayID` 1).
2. Edit the `geometry` argument in the `server-args` line to the desired display resolution. For example,

   ```
   server_args  = :42 -inetd -once -query localhost -geometry 800x600
       -depth 16
   <!--NeedCopy-->
   ```

   The value of the `geometry` parameter can be any valid screen width and height.

3. Save and close the file.
4. Restart the VNC server:

   ```
   /etc/init.d/xinetd restart
   rcxdm restart
   <!--NeedCopy-->
   ```

Check run levels

Red Hat and SUSE Linux VMs use runlevel 5 for graphical startup. This section describes how to verify that your VM starts up in runlevel 5 and how to change this setting.

1. Check `/etc/inittab` to see what the default runlevel is set to. Look for the line that reads:

   ```
   id:n:initdefault:
   <!--NeedCopy-->
   ```

   If *n* is not 5, edit the file to make it so.

2. You can run the command `telinit q ; telinit 5` after this change to avoid having to reboot to switch run levels.

Troubleshoot VM problems

August 22, 2023

XenServer provides two forms of support:

- Free, self-help support on the XenServer website
- Paid-for Support Services, which you can buy from the Support Site.

With XenServer Technical Support, you can open a Support Case online or contact the support center by phone if you experience technical difficulties.

The XenServer Support site hosts several resources that might be helpful to you if you experience unusual behavior, crashes, or other problems. Resources include: Support Forums, Knowledge Base articles, and product documentation.

If you see unusual VM behavior, this section aims to help you solve the problem. This section describes where application logs are located and other information that can help your XenServer Solution Provider track and resolve the issue.

> **Important:**
>
> Follow the troubleshooting information in this section only under the guidance of your XenServer Solution Provider or the Support Team.
>
> Vendor Updates: Keep your VMs up-to-date with operating system vendor-supplied updates. The vendor might have provided fixes for VM crashed and other failures.

VM crashes

If you are experiencing VM crashes, it is possible that a kernel crash dump can help identify the problem. Reproduce the crash, if possible, and follow this procedure. Consult your guest OS vendor for further investigation on this issue.

The crashdump behavior of your VMs can be controlled by using the `actions-after-crash` parameter. The following are the possible values:

Value	Description
`preserve`	Leave the VM in a paused state. (For analysis)
`restart`	No core dump, reboot VM. (This is the default)
`destroy`	No core dump, leave VM halted.

To enable saving of VM crash dumps:

1. On the XenServer host, determine the UUID of the desired VM by running the following command:

   ```
   xe vm-list name-label=<name> params=uuid --minimal
   <!--NeedCopy-->
   ```

2. Change the `actions-after-crash` value by using `xe vm-param-set`; for example, run the following command on dom0:

   ```
   xe vm-param-set uuid=<vm_uuid> actions-after-crash=preserve
   <!--NeedCopy-->
   ```

3. Crash the VM.

   a) Determine the VM's domain ID by running the following command on dom0:

      ```
      xe vm-param-get uuid=<vm_uuid> param-name=dom-id
      <!--NeedCopy-->
      ```

   b) Run the `xl trigger` command in dom0 to trigger the crash:

```
xl trigger <dom_id> nmi
<!--NeedCopy-->
```

Windows VM crashdump behavior

By default Windows crash dumps are put into `%SystemRoot%\Minidump` in the Windows VM itself. You can configure the VMs dump level by following the menu path **My Computer > Properties > Advanced > Startup and Recovery**.

UEFI and Secure Boot problems

How do I change the screen resolution of the XenCenter console on a UEFI-enabled VM?

To change the screen resolution of the XenCenter console on a UEFI-enabled VM:

1. Open the **Windows Settings**
2. Click the **Update & Security** button
3. Under the recovery tab, press the **Restart now** button.
4. Navigate to **Troubleshoot** > **Advanced Options** > **UEFI firmware settings**.
5. Press **Restart**. During restart, the UEFI settings menu loads.
6. Navigate to **Device Manager** > **OVMF Platform Configuration**. This displays the current screen resolution.
7. Press **Enter** to see the screen resolution options.
8. Use the arrow keys to select the desired screen resolution and press **Enter**.
9. Press **F10** to save the changes and confirm your choice.
10. Reboot the VM to see the XenCenter console in an updated screen resolution.

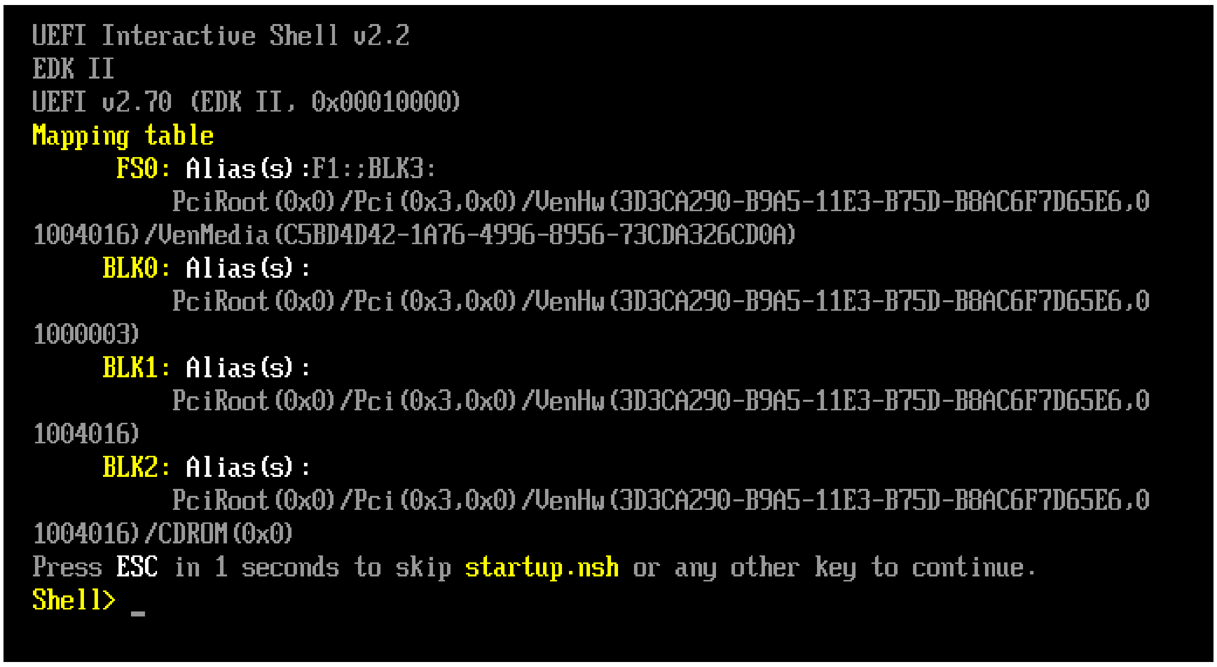
Why can't I create a UEFI Secure Boot VM?

Check that your VM operating system supports UEFI Secure Boot mode. The following operating systems support Secure Boot:

- Windows 10 (64-bit)
- Windows 11 (64-bit)
- Windows Server 2016 (64-bit)
- Windows Server 2019 (64-bit)
- Windows Server 2022 (64-bit)

Why is my UEFI Secure Boot VM failing to start?

If you see the following messages on the console of your UEFI Secure Boot VM and an alert in XenCenter, the Secure Boot process has failed and your VM does not start.



```
UEFI Interactive Shell v2.2
EDK II
UEFI v2.70 (EDK II, 0x00010000)
Mapping table
      FS0: Alias(s):F1:;BLK3:
          PciRoot(0x0)/Pci(0x3,0x0)/VenHw(3D3CA290-B9A5-11E3-B75D-B8AC6F7D65E6,0
1004016)/VenMedia(C5BD4D42-1A76-4996-8956-73CDA326CD0A)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x3,0x0)/VenHw(3D3CA290-B9A5-11E3-B75D-B8AC6F7D65E6,0
1000003)
     BLK1: Alias(s):
          PciRoot(0x0)/Pci(0x3,0x0)/VenHw(3D3CA290-B9A5-11E3-B75D-B8AC6F7D65E6,0
1004016)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x3,0x0)/VenHw(3D3CA290-B9A5-11E3-B75D-B8AC6F7D65E6,0
1004016)/CDROM(0x0)
Press ESC in 1 seconds to skip startup.nsh or any other key to continue.
Shell> _
```

This is usually caused by the installation of unsigned drivers into the VM. Investigate what drivers have been updated or installed since the last successful Secure Boot.

You can disable Secure Boot and start the VM in setup mode to remove the unsigned drivers.

> **Important:**
>
> Before doing this, back up your VM by taking a snapshot.

To change a UEFI Secure Boot VM into a UEFI boot VM, run the following command on the XenServer host that hosts the VM:

```
varstore-sb-state <VM_UUID> setup
```

After you have fixed your VM, run the following command to re-enable Secure Boot:

```
varstore-sb-state <VM_UUID> user
```

Is Secure Boot causing an issue on my VM?

To diagnose whether an issue on your VM is caused by Secure Boot being enabled for the VM, disable Secure Boot and try to reproduce the issue.

To disable Secure Boot, run the following command on the XenServer host that hosts the VM:

```
varstore-sb-state <VM_UUID> setup
```

After you have debugged the issue, you can run the following command to re-enable Secure Boot:

```
varstore-sb-state <VM_UUID> user
```

How do I run Windows debug on a Secure Boot Windows VM?

You cannot run Windows debug on a Secure Boot Windows VM. To run Windows debug on your VM, you can do one of the following things:

- Switch your VM to UEFI boot mode by running the following command:

  ```
  xe vm-param-set uuid=<UUID> platform:secureboot=false
  ```

  Reboot the VM.

  After you have debugged the issue, you can run the following command to re-enable Secure Boot:

  ```
  xe vm-param-set uuid=<UUID> platform:secureboot=auto
  ```

  Reboot the VM.

- Disable Secure Boot by running the following command on the XenServer host that hosts the VM:

  ```
  varstore-sb-state <VM_UUID> setup
  ```

  After you have debugged the issue, you can run the following command to re-enable Secure Boot:

  ```
  varstore-sb-state <VM_UUID> user
  ```

Why are only two NICs showing up for my UEFI-enabled Windows VM?

Even if you set up more than two NICs when you created your UEFI-enabled VM, when the VM first starts you only see two NICs. After the XenServer VM Tools for Windows have been installed in the VM, this information displays correctly.

Why are my emulated devices showing as different types than expected on a UEFI Windows VM?

UEFI Secure Boot VMs use NVME and E1000 for emulated devices. However, when the VM first starts the emulated devices show as different types. After the XenServer VM Tools for Windows have been installed in the VM, this information displays correctly.

Why can't I convert my templates from BIOS mode to UEFI or UEFI Secure Boot mode?

You can only create a UEFI-enabled VM template from a template supplied with XenServer.

Do not use the `xe template-param-set` command for templates that have something installed on them or templates that you created from a snapshot. The boot mode of these snapshots cannot be changed and, if you attempt to change the boot mode, the VM fails to boot.

How do I check UEFI and UEFI Secure Boot variables?

On the XenServer host where the UEFI or UEFI Secure Boot VM is hosted, run the following commands:

```
 varstore-ls
```

This command lists the GUIDs and names of the available variables. Use the GUID and name in the following command:

```
varstore-get <VM\_ID> <GUID> <name> | hexdump -C
```

Why can't I use a 'test'driver with a Secure Boot VM?

If you are also working with a third party to debug and fix issues in their UEFI Secure Boot VM, the third party provide might provide unsigned drivers for test or verification purpose. These drivers do not work in a UEFI Secure Boot VM.

Request a signed driver from the third party. Or you can switch your UEFI Secure Boot VM into setup mode to run with the unsigned driver.

Xentop utility

The xentop utility displays real-time information about a XenServer system and running domains in a semi-graphical format. You can use this tool to investigate the state of the domain associated with a VM.

To run the xentop utility:

1. Connect to the XenServer host over SSH or, in XenCenter, go to the **Console** tab of the host.

2. Run the following command: `xentop`

   The console displays information about the host in a table. The information is periodically refreshed.

Output columns

The xentop utility displays the following columns in the console:

- **NAME** - The name of the domain. "Domain-0"is the XenServer control domain. Other domains belong to the VMs.
- **STATE** - The state of the domain. The state can have one of the following values:
  - d - the domain is dying
  - s –the domain is shutting down
  - b –the domain is blocked
  - c –the domain has crashed
  - p –the domain is paused
  - r –the domain is actively running on one of the CPUs
- **CPU(sec)** - The CPU usage of the domain in seconds
- **CPU(%)** - The CPU usage of the domain as a percentage
- **MEM(k)** - The current memory usage of the domain in KiB
- **MEM(%)** - The current memory usage of the domain as a percentage
- **MAXMEM(k)** - The maximum domain memory usage in KiB
- **MAXMEM(%)** - The maximum domain memory usage as a percentage
- **VCPUS** - The number of virtual CPUs assigned to the domain
- **NETS** - The number of virtual networks used by the domain
- **NETTX(k)** - The amount of total network tx in KiB
- **NETRX(k)** - The amount of total network rx in KiB
- **VBDS** - The number of virtual block devices
- **VBD_OO** - The total number of times that the VBD has encountered an out of requests error. When that occurs, I/O requests for the VBD are delayed.
- **VBD_RD** - The total number of VBD read requests
- **VBD_WR** - The total number of VBD write requests
- **VBD_RSECT** - The VBD read sectors
- **VBD_WSECT** - The VBD write sectors

Xentop parameters

You can use the following parameters to configure the output for the xentop command:

- **-h** - Output the command help for the xentop command.
- **-V** - Output the version of the xentop command.
- **-d** or –**delay=SECONDS** - Set the number of seconds between updates
- **-n** or –**networks** - Output the data for each VIF network associated with a domain
- **-x** or –**vbds** - Output the data for each VBD block device associated with a domain
- **-r** or **repeat-header** - Repeat the table header before each domain
- **-v** or –**vcpus** - Output the data for each vCPU associated with a domain
- **-i** or –**iterations** - Number of iterations (updates) to display before xentop exits
- **-f** or –**full-name** - Output the full domain name instead of a truncated name

You can also configure most of these parameters from within the xentop utility.

High availability

April 17, 2024

High availability is a set of automatic features designed to plan for, and safely recover from issues which take down XenServer hosts or make them unreachable. For example, during physically disrupted networking or host hardware failures.

Overview

High availability ensures that if a host becomes unreachable or unstable, the VMs running on that host are safely restarted on another host automatically. This removes the need for the VMs to be manually restarted, resulting in minimal VM downtime.

When the pool coordinator becomes unreachable or unstable, high availability can also recover administrative control of a pool. High availability ensures that administrative control is restored automatically without any manual intervention.

Optionally, high availability can also automate the process of restarting VMs on hosts which are known to be in a good state without manual intervention. These VMs can be scheduled for restart in groups to allow time to start services. It allows infrastructure VMs to be started before their dependent VMs (for example, a DHCP server before its dependent SQL server).

> **Warnings:**
>
> Use high availability along with multipathed storage and bonded networking. Configure multipathed storage and bonded networking before attempting to set up high availability. Customers who do not set up multipathed storage and bonded networking can see unexpected host reboot behavior (Self-Fencing) when there is an infrastructure instability.
>
> All graphics solutions (NVIDIA vGPU, Intel GVT-d, Intel GVT-G, and vGPU pass-through) can be used in an environment that uses high availability. However, VMs that use these graphics solutions cannot be protected with high availability. These VMs can be restarted on a best-effort basis while there are hosts with the appropriate free resources.

Overcommitting

A pool is overcommitted when the VMs that are currently running cannot be restarted elsewhere following a user-defined number of host failures. Overcommitting can happen if there is not enough free memory across the pool to run those VMs following a failure. However, there are also more subtle changes which can make high availability unsustainable: changes to Virtual Block Devices (VBDs) and networks can affect which VMs can be restarted on which hosts. XenServer cannot check all potential actions and determine if they cause violation of high availability demands. However, an asynchronous notification is sent if high availability becomes unsustainable.

XenServer dynamically maintains a failover plan which details what to do when 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 the high availability configuration. The value of `host failures to tolerate` determines the number of host failures that are allowed while still being able to restart all protected VMs. For example, consider a resource pool that consists of 64 hosts and `host failures to tolerate` is set to 3. In this case, the pool calculates a failover plan that allows any three hosts to fail and then restarts the 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. If changes (for example, the addition of new VMs to the pool) cause the pool to become overcommitted, alerts are sent either via XenCenter or email.

Overcommitment warning

If any attempts to start or resume a VM would cause the pool to become overcommitted, a warning alert is displayed in XenCenter. You can then choose to cancel the operation or proceed anyway. Proceeding causes the pool to become overcommitted and a message is sent to any configured email addresses. This is also available as a message instance through the management API. The amount of memory used by VMs of different priorities is displayed at the pool and host levels.

Host fencing

Sometimes a host can fail due to the loss of network connectivity or when a problem with the control stack is encountered. In such cases, the XenServer host self-fences to ensure that the VMs are not running on two hosts simultaneously. When a fence action is taken, the host restarts immediately and abruptly, causing all VMs running on it to be stopped. The other hosts detect that the VMs are no longer running and then the VMs are restarted according to the restart priorities assigned to them. The fenced host enters a reboot sequence, and when it has restarted it tries to rejoin the resource pool.

> **Note:**
>
> Hosts in clustered pools can also self-fence when they cannot communicate with more than half of the other hosts in the resource pool. For more information, see Clustered pools.

Configuration requirements

To use the high availability feature, you need:

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

  > **Note:**
  >
  > We recommend that you enable high availability only in pools that contain at least 3 XenServer hosts. For more information, see CTX129721 - High Availability Behavior When the Heartbeat is Lost in a Pool.

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

  - 4 MB heartbeat volume: Used to provide a heartbeat.
  - 256 MB metadata volume: To store pool coordinator metadata to be used if there is a failover of the pool coordinator.

  > **Note:**
  >
  > Previously, we recommended using a dedicated NFS or iSCSI storage repository as your high availability heartbeat disk. However, this is only of benefit if the storage repository is not sharing resources on the underlying storage appliance, otherwise it is simply increasing complexity and resource usage in the control domain (dom0) of the host.
  >
  > If your pool is a clustered pool, your heartbeat SR must be a GFS2 SR.
  >
  > Storage attached using either SMB or iSCSI when authenticated using CHAP cannot be used as the heartbeat SR.

- Static IP addresses for all hosts.

  > **Warning:**
  >
  > If the IP address of a host changes while high availability is enabled, high availability assumes that the host's network has failed. The change in IP address can fence the host and leave it in an unbootable state. To remedy this situation, disable high availability using the `host-emergency-ha-disable` command, reset the pool coordinator using `pool-emergency-reset-master`, and then re-enable high availability.

- For maximum reliability, we recommend that you use a dedicated bonded interface as the high availability management network.

For a VM to be protected by high availability, it must be agile. This means the VM:

- Must have its virtual disks on shared storage. You can use any type of shared storage. iSCSI, NFS, or Fibre Channel LUN is only required for the storage heartbeat and can be used for virtual disk storage.
- Can use live migration.
- Does not have a connection to a local DVD drive configured.
- Has its virtual network interfaces on pool-wide networks.

In addition, VMs with an attached vTPM cannot be protected by high availability.

> **Note:**
>
> When high availability is enabled, we strongly recommend using a bonded management interface on the hosts in the pool and multipathed storage for the heartbeat SR.

If you create VLANs and bonded interfaces from the CLI, then they might not be plugged in and active despite being created. In this situation, a VM can appear to be not agile and it is not protected by high availability. You can use the CLI `pif-plug` command to bring up the VLAN and bond PIFs 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. This command analyzes its placement constraints, and you can take remedial action if necessary.

Restart configuration settings

Virtual machines can be considered protected, best-effort, or unprotected by high availability. The value of `ha-restart-priority` defines whether a VM is treated as protected, best-effort, or unprotected. The restart behavior for VMs in each of these categories is different.

Protected

High availability guarantees to restart a protected VM that goes offline or whose host goes offline, provided the pool isn't overcommitted and the VM is agile.

If a protected VM cannot be restarted when a host fails, high availability attempts to start the VM when there is extra capacity in a pool. Attempts to start the VM when there is extra capacity might now succeed.

`ha-restart-priority` Value: `restart`

Best-effort

If the host of a best-effort VM goes offline, high availability attempts to restart the best-effort VM on another host. It makes this attempt only after all protected VMs have been successfully restarted. High availability makes only one attempt to restart a best-effort VM. If this attempt fails, high availability does not make further attempts to restart the VM.

`ha-restart-priority` Value: `best-effort`

Unprotected

If an unprotected VM or the host it runs on is stopped, high availability does not attempt to restart the VM.

`ha-restart-priority` Value: Value is an empty string

> **Note:**
>
> High availability never stops or migrates a running VM to free resources for a protected or best-effort VM to be restarted.

If the pool experiences host failures and the number of tolerable failures drops to zero, the protected VMs are not guaranteed to restart. In such cases, a system alert is generated. If another failure occurs, all VMs that have a restart priority set behave according to the best-effort behavior.

Start order

The start order is the order in which XenServer high availability attempts to restart protected VMs when a failure occurs. The values of the `order` property for each of the protected VMs determines the start order.

The `order` property of a VM is used by high availability and also by other features that start and shut down VMs. Any VM can have the `order` property set, not just the VMs marked as protected for high availability. However, high availability uses the `order` property for protected VMs only.

The value of the `order` property is an integer. The default value is 0, which is the highest priority. Protected VMs with an `order` value of 0 are restarted first by high availability. The higher the value of the `order` property, the later in the sequence the VM is restarted.

You can set the value of the `order` property of a VM by using the command-line interface:

```
xe vm-param-set uuid=<vm uuid> order=<int>
<!--NeedCopy-->
```

Or in XenCenter, in the **Start Options** panel for a VM, set **Start order** to the required value.

Enable high availability on your XenServer pool

You can enable high availability on a pool by using either XenCenter or the command-line interface (CLI). In either case, you specify a set of priorities that determine which VMs are given the highest restart priority when a pool is overcommitted.

> **Warnings:**
>
> - When you enable high availability, some operations that compromise the plan for restarting VMs (such as removing a host from a pool) might be disabled. You can temporarily disable high availability to perform such operations, or alternatively, make VMs protected by high availability unprotected.
> - If high availability is enabled, you cannot enable clustering on your pool. Temporarily disable high availability to enable clustering. You can then enable high availability on your clustered pool. Some high availability behavior, such as self-fencing, is different for clustered pools. For more information, see Clustered pools.

Enable high availability by using the CLI

1. Verify that you have a compatible Storage Repository (SR) attached to your pool. iSCSI, NFS, or Fibre Channel SRs are compatible. For information about how to configure such a storage repository using the CLI, see Manage storage repositories.
2. For each VM you want to protect, set a restart priority and start order. You can set the restart priority as follows:

   ```
   xe vm-param-set uuid=<vm uuid> ha-restart-priority=restart order=1
   <!--NeedCopy-->
   ```

3. Enable high availability on the pool, and optionally, specify a timeout:

   ```
   xe pool-ha-enable heartbeat-sr-uuids=<sr uuid> ha-config:timeout=<
       timeout in seconds>
   <!--NeedCopy-->
   ```

   Alternatively, you can set a default timeout for your pool. For more information about how to set a timeout, see Configure high availability timeout.

4. Run the command `pool-ha-compute-max-host-failures-to-tolerate`. 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
   <!--NeedCopy-->
   ```

   The number of failures to tolerate determines when an alert is sent. The system recomputes a failover plan as the state of the pool changes. It uses this computation to identify the pool capacity 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 `ha-host-failures-to-tolerate` parameter. The value must be less than or equal to the computed value:

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

Configure high availability timeout

The timeout is the period during which networking or storage is not accessible by the hosts in your pool. If any XenServer host is unable to access networking or storage within the timeout period, it can self-fence and restart. The default timeout is 60 seconds. However, you can change this value by using the following command.

Set a default high availability timeout for your pool:

```
xe pool-param-set uuid=<pool uuid> other-config:default_ha_timeout=<
    timeout in seconds>
<!--NeedCopy-->
```

If you enable high availability by using XenCenter instead of the xe CLI, this default still applies.

Alternatively, you can set timeout when you enable high availability:

```
xe pool-ha-enable heartbeat-sr-uuids=<sr uuid> ha-config:timeout=<
    timeout in seconds>
<!--NeedCopy-->
```

Note that if you set timeout when enabling high availability, it only applies to that specific enablement. Therefore, if you disable and then reenable high availability later, the high availability feature reverts to using the default timeout.

Remove high availability protection from a VM by using the CLI

To disable high availability features for a VM, use the `xe vm-param-set` command to set the `ha-restart-priority` parameter to be an empty string. Setting the `ha-restart-priority` parameter does not clear the start order settings. You can enable high availability for a VM again by setting the `ha-restart-priority` parameter to `restart` or `best-effort` as appropriate.

Recover an unreachable host

If for some reason, a host cannot access the high availability state file, it is possible that a host might become unreachable. To recover your XenServer installation, you might have to disable high availability using the `host-emergency-ha-disable` command:

```
xe host-emergency-ha-disable --force
<!--NeedCopy-->
```

If the host was the pool coordinator, it starts up as normal with high availability disabled. Pool members reconnect and automatically disable high availability. If the host was a pool member and cannot contact the pool coordinator, you might have to take one of the following actions:

- Force the host to reboot as a pool coordinator (`xe pool-emergency-transition-to-master`)

  ```
  xe pool-emergency-transition-to-master uuid=<host uuid>
  <!--NeedCopy-->
  ```

- Tell the host where the new pool coordinator is (`xe pool-emergency-reset-master`):

  ```
  xe pool-emergency-reset-master master-address=<new pool
      coordinator hostname>
  <!--NeedCopy-->
  ```

When all hosts have successfully restarted, re-enable high availability:

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

Shutting down a host when high availability is enabled

Take special care when shutting down or rebooting a host to prevent the high availability mechanism from assuming that the host has failed. To shut down a host cleanly when high availability is enabled, disable the host, evacuate the host, and finally shutdown the host by using either XenCenter or the CLI. To shut down a host in an environment where high availability is enabled, run these commands:

```
xe host-disable host=<host name>
xe host-evacuate uuid=<host uuid>
xe host-shutdown host=<host name>
<!--NeedCopy-->
```

Shut down a VM protected by high availability

When a VM is protected under a high availability plan and set to restart automatically, it cannot be shut down while this protection is active. To shut down a VM, first disable its high availability protection and then run the CLI command. XenCenter offers you a dialog box to automate disabling the protection when you select 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 high availability failure conditions. The automatic restart helps ensure that operator error doesn't result in a protected VM being left shut down accidentally. If you want to shut down this VM, disable its high availability protection first.

Disaster recovery and backup

November 21, 2023

The XenServer Disaster Recovery (DR) feature allows you to recover virtual machines (VMs) and vApps from a failure of hardware which destroys a whole pool or site. For protection against single host failures, see High availability.

> **Note:**
>
> You must be logged on with your *root* account or have the role of *Pool Operator* or higher to use the DR feature.

Understanding XenServer DR

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

The **Disaster Recovery** settings in XenCenter can be used to query the storage and import selected VMs and vApps to a recovery pool during a disaster. When the VMs are running in the recovery pool, the recovery pool metadata is also replicated. The replication of the pool metadata allows any changes in VM settings to be populated back to the primary pool when the primary pool recovers. Sometimes, information for the same VM can be in several 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 to. If XenCenter finds that the VM information is present in two or more places, it ensures that it uses only the most recent information.

The Disaster Recovery feature can be used with XenCenter and the xe CLI. For CLI commands, see Disaster recovery commands.

> **Tip:**
>
> You can also use the Disaster Recovery settings to run test failovers for non-disruptive testing of your disaster recovery system. In a test failover, all the steps are the same as failover. However, the VMs and vApps are not started up after they have been recovered to the disaster recovery site. When the test is complete, cleanup is performed to delete 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 information is required 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 change the VM configuration. For VMs in a pool, a copy of this metadata is stored on every host in the pool.

In a DR environment, VMs are recreated on a secondary site using the pool metadata and configuration information about all 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 VM startup options –start order, delay interval, high availability, and restart priority. The VM startup options are used when restarting the VM in a high availability or DR environment. For example, when recovering VMs during disaster recovery, VMs within a vApp are restarted in the DR pool in the order specified in the VM metadata, and using the specified delay intervals.

DR infrastructure requirements

Set up the appropriate DR infrastructure at both the primary and secondary sites to use XenServer DR.

- Storage used for pool metadata *and* the virtual disks used by the VMs must be replicated from the primary (production) environment to a backup environment. Storage replication such as using mirroring varies between devices. Therefore, consult your storage solution vendor to handle Storage replication.
- After the VMs and vApps that you recovered to a pool on your DR site are up and running, the SRs containing the DR pool metadata and virtual disks must be replicated. Replication allows the recovered VMs and vApps to be restored back to the primary site *(failed back)* when the primary site is back online.
- The hardware infrastructure at your DR site does not have to match the primary site. However, the XenServer environment must be at the same release and patch level.
- The hosts and pools at the secondary site must have the same license edition as those at the primary site. These XenServer licenses are in addition to those assigned to hosts at the primary site.
- Sufficient resources must be configured in the target pool to allow all the failed over VMs to be recreated and started.

> **Warning:**
>
> The Disaster Recovery settings do 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 you use these features, you must disable the mirror functionality ("mirror is broken") before restarting VMs on the recovery site.

Deployment considerations

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` parameters. Recovering VMs and vApps from replicated storage is easier if the names of SRs capture how VMs and vApps are mapped to SRs, and SRs to LUNs.
- Arrange replication of the LUNs.
- Enable pool metadata replication to one or more SRs on these LUNs.
- Ensure that the SRs you are replicating the primary pool metadata to are attached to only one pool.

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 want to recover VM data from are not attached to any other pool, or corruption can 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.

- Resynchronize any storage mirrors.
- On the recovery site, cleanly shut down 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 in the previous section, 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.

Enable Disaster Recovery

May 22, 2023

This section describes how to enable Disaster Recovery in XenCenter. Use the **Configure DR** option to identify storage repositories where the pool metadata, configuration information about all the VMs and vApps in the pool is stored. The metadata is updated whenever you change the VM or vApp configuration within the pool.

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

Before you begin, ensure that the SRs used for DR are attached only to the pool at the primary site. SRs used for DR must not be attached to the pool at the secondary site.

To configure Disaster Recovery, complete the following steps:

1. On the primary site, select the pool that you want to protect. From the **Pool** menu, point to **Disaster Recovery**, and then select **Configure**.
2. Select up to 8 SRs where the pool metadata can be stored. A small amount of space is required on this storage for a new LUN which contains 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. Select **OK**. Your pool is now protected.

Recover VMs and vApps during a 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, select **Disaster Recovery** and then **Disaster Recovery Wizard**.

   The Disaster Recovery wizard displays three recovery options: **Failover**, **Failback**, and **Test Failover**. To recover on to your secondary site, select **Failover** and then select **Next**.

   > **Warning:**
   >
   > If you use Fibre Channel shared storage with LUN mirroring to replicate data to the secondary site, break the mirroring before attempting to recover VMs. Mirroring must be broken to ensure 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, select **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 type the target host, IQN, and LUN details.

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

3. Select the VMs and vApps that you want to recover. Select the appropriate **Power state after recovery** option to specify whether you want the wizard to start them up automatically when they have been recovered. Alternatively, you can start them up manually after failover is complete.

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

4. The wizard performs several 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 select **Attach SR** on this page to find and attach the relevant SR.

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

5. A progress page displays the result of the recovery process for each VM and vApp. The Failover process exports the metadata for VMs and vApps from the replicated storage. Therefore, the time taken for Failover depends on the VMs and vApps you recover. The VMs and vApps are recreated in the primary pool, and the SRs containing the virtual disks are attached to the recreated VMs. If specified, the VMs are started.

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

When the primary site is available, work through the Disaster Recovery wizard and select **Failback** to return to running your VMs on that site.

Restore VMs and vApps to the primary site after disaster (Failback)

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

1. In XenCenter select the primary pool, and on the Pool menu, select **Disaster Recovery** and then **Disaster Recovery Wizard**.

The Disaster Recovery wizard displays three recovery options: **Failover**, **Failback**, and **Test Failover**. To restore VMs and vApps to your primary site, select **Failback** and then select **Next**.

> **Warning:**
>
> When you use Fibre Channel shared storage with LUN mirroring to replicate data to the primary site, break the mirroring before attempting to restore VMs. Mirroring must be broken to ensure that the primary 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 type the target host, IQN, and LUN details.

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

3. Select the VMs and vApps that you want to restore. Select the appropriate **Power state after recovery** option to specify whether you want the wizard to start them up automatically when they have been recovered. Alternatively, you can start them up manually after failback is complete.

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

4. The wizard performs several 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 select **Attach SR on this page** to find and attach the relevant SR.

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

5. A progress page displays the result of the recovery process for each VM and vApp. The Failback process exports the metadata for VMs and vApps from the replicated storage. Therefore, Failback can take some time depending on the number of VMs and vApps you are restoring. The VMs and vApps are recreated in the primary pool, and the SRs containing the virtual disks are attached to the recreated VMs. If specified, the VMs are started.

6. When the failback is complete, select **Next** to see the summary report. Select **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, the steps are the same as for failover. However, 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 deleted. After initial DR configuration, and after you make significant configuration changes in a DR-enabled pool, verify that failover works correctly by performing a test failover.

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

   The Disaster Recovery wizard displays three recovery options: **Failover**, **Failback**, and **Test Failover**. To test your disaster recovery system, select **Test Failover** and then select **Next**.

   > **Note:**
   >
   > If you use Fibre Channel shared storage with LUN mirroring to replicate data to the secondary site, break the mirroring before attempting to recover data. Mirroring must be broken to ensure 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, select **Find Storage Repositories** and then 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 type the target host, IQN, and LUN details in the box.

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

3. Select the VMs and vApps that you want to recover then select **Next** to progress to the next page and begin failover prechecks.

4. Before beginning the test failover, the wizard performs several 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 select Attach SR on this page to find and attach the relevant SR.
   - **Check that high availability is not enabled on the target DR pool**. High availability must be disabled on the secondary pool to avoid having the same VMs running on both the primary and DR pools. High availability must be disabled to ensure that it does not start the

recovered VMs and vApps automatically after recovery. To disable high availability on the secondary pool, you can simply select **Disable HA** on the page. If high availability is disabled at this point, it is enabled again automatically at the end of the test failover process.

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

5. A progress page displays the result of the recovery process for each VM and vApp. The Failover process recovers metadata for the VMs and vApps from the replicated storage. Therefore, Failover can take some time depending on the number of VMs and vApps you are recovering. 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 do not start up on the secondary site during a test failover.

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

   - VMs and vApps that were recovered during the test failover are deleted.
   - Storage that was recovered during the test failover is detached.
   - If high availability on the DR pool was disabled at the prechecks stage to allow the test failover to take place, it is re-enabled automatically.

   The progress of the cleanup process appears on the wizard.

7. Select **Finish** to close the wizard.

vApps

May 22, 2023

A vApp is logical group of one or more related Virtual Machines (VMs). vApps can be started up as a single entity when there is a disaster. When a vApp is started, the VMs contained within the vApp start in a user predefined order. The start order allows VMs which depend upon one another to be automatically sequenced. An administrator no longer has to manually sequence the startup of dependent VMs when a whole service requires restarting. For example, during a software update. The VMs within the vApp do not have to reside on one host and are distributed within a pool using the normal rules. The vApp feature is useful in the Disaster Recovery (DR) situation. In a DR scenario, an Administrator may group all VMs on the same Storage Repository, or which relate to the same Service Level Agreement (SLA).

To group VMs together in a vApp follow the procedure:

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

   You can choose any name you like, but an informative name is best. Although we recommend you to avoid having multiple vApps using the same name, it is not a requirement. XenCenter does not enforce any constraints regarding unique vApp names. It is not necessary to use quotation marks for names that include spaces.
3. Select which VMs to include in the new vApp, and then click **Next**.

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

   Start Order: Specifies the order in which individual VMs are started within the vApp, allowing certain VMs to be restarted before others. VMs with a start order value of 0 (zero) are started first. VMs with a start order value of 1 are started next, and then the VMs with a value of 2, and so on.

   Attempt to start next VM after: 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.
5. You can review the vApp configuration on the final page. Click **Previous** to go back and change any settings, or **Finish** to create the vApp.

> **Note:**
>
> A vApp can span multiple hosts in a single pool, but cannot span across several pools.

Manage vApps in XenCenter

The **Manage vApps** setting in XenCenter allows you to create, delete, and change vApps. It also enables you to start and shut down 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. For more information, see vApps in the XenCenter documentation.

Back up and restore hosts and VMs

May 22, 2023

Whenever possible, 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 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 TFTP server and appropriate answer files for this purpose. For more information, see Network boot installations.

We recommend that you use a backup solution offered by one of our certified partners. For more information, see Citrix Ready Marketplace.

XenServer Premium Edition customers can take advantage of the faster changed block only backup. For more information, see the changed block tracking documentation.

We recommend that you frequently perform as many of the following backup procedures as possible to recover from possible server and software failure.

To back up pool metadata:

1. Run the command:

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

2. To restore the database, run the command:

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

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

To back up host configuration and software:

1. Run the command:

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

> **Notes:**
>
> - Do not create the backup in the control domain.
> - The backup procedure can create a large backup file.
> - To complete a restore, you must reboot to the original install CD.
> - This data can only be restored to the original machine.

To back up 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
   <!--NeedCopy-->
   ```

> **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:**
>
> The backup process can take longer to complete as it backs up all of the VM data.

To back up VM metadata only:

Run the command:

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

Back 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 SRs, this database forms the complete view of all VMs available across the pool. Therefore it is important to understand how to back up this database to recover from physical hardware failure and other disaster scenarios.

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

Back up single host installations

Use the CLI to back up 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 contains 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 you restore the pool database, some VMs may still be registered as being `Suspended`. However, if the storage repository with the suspended memory state defined in the `suspend-VDI-uuid` field, is a local SR, then the SR may not be available as the host has been reinstalled. To reset these VMs back to the `Halted` state so that they can start 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 preserves UUIDs of the hosts restored using this method. If you restore to a different physical machine while the original XenServer host is still running, duplicate UUIDs may be present. As a result, XenCenter refuses to connect to the second XenServer host. Pool database backup is not the recommended mechanism for cloning physical hosts. Use the automated installation support instead. For more information, see Install.

Back up pooled installations

In a pool scenario, the pool coordinator provides an authoritative database that is synchronously mirrored to all the pool member hosts. This process provides a level of built-in redundancy to a pool. Any pool member can replace the pool coordinator because each pool member has an accurate version of the pool database. For more information on how to transition a member into becoming a pool coordinator, see Hosts and resource pools.

This level of protection may not be sufficient. For example, when shared storage containing the VM data is backed up in multiple sites, but the local server storage (containing the pool metadata) is not. To re-create a pool given a set of shared storage, you must first back up the `pool-dump-database` file on the pool coordinator host, and archive this file. To restore this backup later on a brand new set of hosts:

1. Install a fresh set of XenServer hosts from the installation media, or if applicable, network boot from your TFTP server.
2. Use the `xe pool-restore-database` on the host designated to be the new pool coordinator.
3. Run the `xe host-forget` command on the new pool coordinator to remove the old member machines.
4. Use the `xe pool-join` command on the member hosts to connect them to the new pool.

Back 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:**
>
> The privileged control domain is best left as installed, without customizing it with other packages. We recommend that you set up a network boot environment to install XenServer cleanly

> from the XenServer media as a recovery strategy. Typically, you do not need to back up the control domain, but we recommend that you save the pool metadata (see Back up virtual machine metadata). Consider this backup method as 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. 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 selecting to restore the appropriate backup.

After completing the steps in the previous section and rebooting the host, ensure that the VM metadata is restored to a consistent state. Run `xe pool-restore-database` on `/var/backup/pool-database-${ DATE }` to restore the VM metadata. 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 your XenServer host:

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

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

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

To restore a running XenServer host:

1. If you want to restore your 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
   <!--NeedCopy-->
   ```

   This command restores the compressed image back to the hard disk of the XenServer host which runs this command (not the host on which `filename` resides). In this context, "restore" may be a misnomer, as the word usually suggests that the backed-up state has been put fully in place. The restore command only unpacks the compressed backup file and restores it to its normal form. However, 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.

3. Finally, restore the VM metadata using the following command:

```
xe pool-restore-database file-name=/var/backup/pool-database-* -h
    hostname -u root -pw password
<!--NeedCopy-->
```

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

To restart a crashed XenServer host:

If your XenServer host has crashed and is not reachable, use the XenServer installation CD to do an upgrade install. When the upgrade install is complete, reboot the machine and ensure that your host is reachable with XenCenter or remote CLI.

Then proceed with backing up XenServer hosts as describes in this section.

Back up VMs

We recommend that you use a backup solution offered by one of our certified partners. For more information, see Citrix Ready Marketplace.

XenServer Premium Edition customers can take advantage of the faster changed block only backup. For more information, see the Citrix blog about Changed Block Tracking backup APIs.

VM snapshots

March 13, 2024

XenServer provides a convenient mechanism that can take a snapshot of a VM storage and metadata at a given time. Where necessary, I/O 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. However, for the LVM-based storage types the following requirements must be met:

- If the storage repository was created on a previous version of XenServer, it must have been upgraded
- The volume must be in the default format (you cannot take a snapshot of `type=raw` volumes)

The snapshot operation is a two-step process:

- Capturing metadata as a template.
- Creating a VDI snapshot of the disks.

The following types of VM snapshots are supported: regular and snapshot with memory.

Regular snapshots

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

Snapshots with memory

In addition to saving the VMs memory (storage) and metadata, snapshots with memory also save the VMs state (RAM). This feature can be useful when you upgrade or patch software, but you also want the option to revert to the pre-change VM state (RAM). Reverting 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 by using the management API, the xe CLI, or XenCenter.

Create a VM snapshot

Before taking a snapshot, see the following information about any special operating system-specific configuration and considerations:

- Prepare to clone a Windows VM by using Sysprep
- Prepare to clone a Linux VM

First, 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`command to take a snapshot of a VM.

```
xe vm-snapshot vm=vm uuid new-name-label=vm_snapshot_name
<!--NeedCopy-->
```

Create 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
<!--NeedCopy-->
```

When XenServer has completed creating the snapshot with memory, its UUID is displayed.

For example:

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

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

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

To list all of the snapshots on your XenServer pool

Run the `snapshot-list` command:

```
xe snapshot-list
<!--NeedCopy-->
```

This command lists all of the snapshots in the XenServer pool.

To list the snapshots on a particular VM

Get the UUID of the particular VM by running the `vm-list` command.

```
xe vm-list
<!--NeedCopy-->
```

This command 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 2016 (1)
power-state ( RO): halted

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

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

Locate the required VM and then enter the following:

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

For example:

```
xe snapshot-list snapshot-of=2d1d9a08-e479-2f0a-69e7-24a0e062dd35
<!--NeedCopy-->
```

This command 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
<!--NeedCopy-->
```

Restore 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:

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

   ```
   xe snapshot-list
   <!--NeedCopy-->
   ```

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

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

   For example:

```
xe snapshot-revert snapshot-uuid=b3c0f369-59a1-dd16-ecd4-
    a1211df29886
<!--NeedCopy-->
```

After reverting a VM to a checkpoint, the VM is suspended.

> **Notes:**
>
> - If there's insufficient disk space available to thickly provision the snapshot, you cannot restore to the snapshot until the current disk's state has been freed. If this issue occurs, retry the operation.
> - It is possible to revert to any snapshot. Existing snapshots and checkpoints are not deleted during the revert operation.

Delete a snapshot

Ensure that you have the UUID of the checkpoint or snapshot that you want 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
   <!--NeedCopy-->
   ```

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

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

3. This command alerts you to the VM and VDIs that are 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
<!--NeedCopy-->
```

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

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

For example:

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

Snapshot templates

Create a template from a snapshot

You can create a VM template from a snapshot. However, its memory state is removed.

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
   <!--NeedCopy-->
   ```

   For example:

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

   > **Note:**
   >
   > This command 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
   <!--NeedCopy-->
   ```

   This command lists all of the templates on the XenServer host.

Export 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. This file has a `.xva` file name extension.

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

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

For example:

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

The VM export/import feature can be used in various different ways:

- As a convenient backup facility for your VMs. An exported VM file can be used to recover an entire VM in a disaster scenario.
- 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 host.

For more information about the use of templates, see Create VMs and also the Managing VMs article in the XenCenter documentation.

Scheduled snapshots

The Scheduled Snapshots 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 individual VMs. Scheduled Snapshots work by having pool-wide snapshot schedules for selected VMs in the pool. When a snapshot schedule is enabled, Snapshots of the specified VM are taken at the scheduled time each hour, day, or week. Several Scheduled Snapshots may be enabled in a pool, covering different VMs and with different schedules. A VM can be assigned to only one snapshot schedule at a time.

XenCenter provides a range of tools to help you use this feature:

- To define a Scheduled Snapshot, use the **New snapshot schedule** wizard.
- To enable, disable, edit, and delete Scheduled Snapshots for a pool, use the **VM Snapshot Schedules** dialog box.
- To edit a snapshot schedule, open its **Properties** dialog box from the **VM Snapshot Schedules** dialog box.
- To revert a VM to a scheduled snapshot, select the snapshot on the **Snapshots** tab and revert the VM to it.

For more information, see Scheduled Snapshots in the XenCenter documentation.

Cope with machine failures

May 22, 2023

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 Backup.

Member failures

In the absence of HA, pool coordinator nodes detect the failures of members by receiving regular heartbeat messages. If no heartbeat has been received for 600 seconds, the pool coordinator assumes the member is dead. There are two ways to recover from this problem:

- Repair the dead host (for example, by physically rebooting it). When the connection to the member is restored, the pool coordinator marks the member as alive again.
- Shut down the host and instruct the pool coordinator 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 are marked as offline and can be restarted on other XenServer hosts.

  It is important to ensure that the XenServer host is actually offline, otherwise VM data corruption might occur.

  Do not to split your pool into multiple pools of a single host by using `xe host-forget`. This action might result in them all mapping the same shared storage and corrupting VM data.

> **Warning:**
>
> - If you are going to use the forgotten host as an active 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 might 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 vm-reset-powerstate for more details.

> **Warning:**
>
> Incorrect use of this command can lead to data corruption. Only use this command if necessary.

Before you can start VMs on another XenServer host, you are also required to release the locks on VM storage. Only on host at a time can use each disk in an SR. 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 coordinator 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 pool coordinator at the time of the crash. (The SR pool coordinator is the pool coordinator or a XenServer host using local storage.)

> **Warning:**
>
> Be sure that the host is down before running 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 `resetvdis.py` script, then you receive the following error message: `VDI <UUID> already attached RW`.

Pool coordinator failures

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

1. If HA is enabled, another pool coordinator is elected automatically.
2. If HA is not enabled, each member waits for the pool coordinator to return.

If the pool coordinator comes back up at this point, it re-establishes communication with its members, and operation returns to normal.

If the pool coordinator is dead, choose one of the members and run the command `xe pool-emergency-transition-to-master` on it. Once it has become the pool coordinator, run the command `xe pool-recover-slaves` and the members now point to the new pool coordinator.

If you repair or replace the host that was the original pool coordinator, you can simply bring it up, install the XenServer 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 host the pool coordinator.

When a member XenServer host is transitioned to being a pool coordinator, check that the default pool storage repository is set to an appropriate value. This check 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 must recreate the pool database from scratch. Be sure to regularly back up your pool-metadata using the `xe pool-dump-database` CLI command (see `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 pool coordinator, restore the pool database from your backup using the `xe pool-restore-database` command (see pool-restore-database).
3. Connect to the pool coordinator by 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.

Cope with failure due to configuration errors

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

1. Run the following command to restore host software and configuration:

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

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 from the following list to recover.

> **Warning:**
>
> Any VMs running on a previous member (or the previous host) which have failed are still marked as `Running` in the database. This behavior 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.

To replace a failed pool coordinator with a still running member:

1. Run the following commands:

   ```
   xe pool-emergency-transition-to-master
   xe pool-recover-slaves
   <!--NeedCopy-->
   ```

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
   <!--NeedCopy-->
   ```

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

2. If the target machine has a different view of the storage than the original machine, modify the storage configuration using the `pbd-destroy` command. Next use the `pbd-create` command to recreate storage configurations. See pbd commands for documentation of these commands.

3. If you have created a 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 following command:

   ```
   xe vm-import filename=backup metadata=true
   <!--NeedCopy-->
   ```

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

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

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

3. Restart all VMs.

Workload Balancing

March 14, 2024

> **Notes:**
>
> - Workload Balancing is available for XenServer Premium Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.
> - Workload Balancing 8.3.0 and later are compatible with XenServer 8. If you perform a rolling pool upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8, you cannot use Workload Balancing 8.2.2 with your XenServer 8 pools. Update the Workload Balancing virtual appliance to 8.3.0 before performing the rolling pool upgrade. You can download the latest version of the Workload Balancing virtual appliance from the XenServer Downloads page.

Workload Balancing is a XenServer Premium Edition component, packaged as a virtual appliance, that provides the following features:

- Create reports about virtual machine (VM) performance in your XenServer environment
- Evaluate resource utilization and locates VMs on the best possible hosts in the pool for their workload's needs
- Balance VM workloads across hosts in a XenServer resource pool
- Determine the best host on which to start a VM
- Determine the best host on which to resume a VM that you powered off
- Determine the best host to move a VM to when a host fails
- Determine the optimal server for each of the host's VMs when you put a host into or take a host out of maintenance mode

Depending on your preference, Workload Balancing can accomplish these tasks automatically or prompt you to accept its rebalancing and placement recommendations. You can also configure Workload Balancing to power off hosts automatically at specific times of day. For example, configure your hosts to switch off at night to save power.

Workload Balancing can send notifications in XenCenter regarding the actions it takes. For more information on how to configure the alert level for Workload Balancing alerts by using the xe CLI, see Set alert level for Workload Balancing alerts in XenCenter.

Workload Balancing functions by evaluating the use of VMs across a pool. When a host exceeds a performance threshold, Workload Balancing relocates the VM to a less-taxed host in the pool. To rebalance workloads, Workload Balancing moves VMs to balance the resource use on hosts.

To ensure that the rebalancing and placement recommendations align with your environment's needs, you can configure Workload Balancing to optimize workloads in one of the following ways:

- To maximize resource performance
- To maximize the number of VMs that fit on hosts

These optimization modes can be configured to change automatically at predefined times or stay the same always. For extra granularity, fine-tune the weighting of individual resource metrics: CPU, network, disk, and memory.

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

As Workload Balancing captures performance data, you can also use this component to generate reports, known as Workload Reports, about your virtualized environment. For more information, see Generate workload reports.

Workload Balancing basic concepts

When VMs are running, they consume computing resources on the physical host. These resources include CPU, Memory, Network Reads, Network Writes, Disk Reads, and Disk Writes. Some VMs, depending on their workload, might consume more CPU resources than other VMs on the same host. Workload is defined by the applications running on a VM and their user transactions. The combined resource consumption of all VMs on a host reduces the available resources on the host.

Workload Balancing captures data for resource performance on VMs and physical hosts and stores it in a database. Workload Balancing uses this data, combined with the preferences you set, to provide optimization and placement recommendations.

Optimizations are a way in which hosts are "improved"to align with your goals: Workload Balancing makes recommendations to redistribute the VMs across hosts in the pool to increase either performance or density. When Workload Balancing is making recommendations, it makes them in light of its goal: to create balance or harmony across the hosts in the pool. If Workload Balancing acts on these recommendations, the action is known as an optimization.

When Workload Balancing is enabled, XenCenter provides star ratings to indicate the optimal hosts for starting a VM. These ratings are also provided:

- When you want to start the VM when it is powered off
- When you want to start the VM when it is suspended
- When you want to migrate the VM to a different host (Migrate and Maintenance Mode)

Within a Workload Balancing context:

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

Workload Balancing does not conflict with settings you already specified for High Availability: these features are compatible.

What's new in Workload Balancing

March 13, 2024

The latest version of the Workload Balancing virtual appliance is version 8.3.0. You can download this version of the Workload Balancing virtual appliance from the XenServer Downloads page.

What's new in 8.3.0

Released Feb 23, 2023

This update includes the following improvements:

- You can now set the alert level for Workload Balancing alerts in XenCenter by using the Management API.

This update includes changes to the WLB database. Ensure that you use the provided migration script when you update your WLB to this version. For more information about using the migration script, see Migrate data from an existing virtual appliance.

Fixed issues in 8.3.0

This update fixes the following issues:

- During the Workload Balancing maintenance window, Workload Balancing is unable to provide placement recommendations. When this situation occurs, you see the error: "4010 Pool discovery has not been completed. Using original algorithm."The Workload Balancing maintenance window is less than 20 minutes long and by default is scheduled at midnight.

- For a Workload Balancing virtual appliance version 8.2.2 and later that doesn't use LVM, you cannot extend the available disk space.
- Due to an unresponsive API call, Workload Balancing is sometimes blocked during pool discovery.
- In XenCenter, the date range and some timestamps shown on the Workload Balancing Pool Audit Report are incorrect.
- In XenCenter, some strings are not displaying correctly for Workload Reports.
- If the Workload Balancing virtual appliance is running for a long time, it is shut down by the operating system for consuming a lot of memory.
- The database fails to auto-restart after the Workload Balancing virtual appliance experiences an abnormal shutdown.

Earlier releases

This section lists features in previous releases along with their fixed issues. These earlier releases are superseded by the latest version of the Workload Balancing virtual appliance. Update to the latest version of the Workload Balancing virtual appliance when it is available.

XenCenter 8.2.2

Released Sep 30, 2021

This update includes changes to the WLB database. Ensure that you use the provided migration script when you update your WLB to this version. For more information about using the migration script, see Migrate data from an existing virtual appliance.

Fixed issues This update includes fixes for the following issues:

- The Workload Balancing database can grow very fast and fill the disk.
- A race condition can sometimes cause records to be duplicated in the WLB database. When this occurs, the user might see the error: "WLB received an unknown exception".

XenCenter 8.2.1

Released Sep 15, 2020

This update includes the following improvements:

- The migration script now enables you to migrate your Workload Balancing database from the Workload Balancing virtual appliance 8.0.0 (which was provided with Citrix Hypervisor 8.0 and 8.1) to the Workload Balancing virtual appliance 8.2.1 provided with Citrix Hypervisor 8.2.

For more information about using the migration script, see Migrate data from an existing virtual appliance.

Fixed issues This update includes fixes for the following issues:

- When multiple VMs start at the same time, Workload Balancing recommends balancing the VMs placement on all hosts in the pool evenly. However, sometimes Workload Balancing might recommend to put many VMs on the same XenServer host. This issue occurs when Workload Balancing gets late feedback from XAPI about VM placement.

Get started with Workload Balancing

March 15, 2024

You can configure the Workload Balancing virtual appliance in just a few steps:

1. Review the prerequisite information and plan your Workload Balancing usage.
2. Download the Workload Balancing virtual appliance.
3. Import the Workload Balancing virtual appliance into XenCenter.
4. Configure Workload Balancing virtual appliance from the virtual appliance console.
5. (Optional) If you already have a previous version of Workload Balancing installed, you can migrate data from an existing virtual appliance.

   > **Note:**
   >
   > If you perform a rolling pool upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8, you cannot use Workload Balancing 8.2.2 and earlier with your XenServer 8 pools. Update your Workload Balancing virtual appliance to version 8.3.0 before performing the rolling pool upgrade. You can download the latest version of the Workload Balancing virtual appliance from the XenServer Downloads page.

6. Connect your pool to the Workload Balancing virtual appliance by using XenCenter.

   The Workload Balancing tab only appears in XenCenter if your pool has the required license to use Workload Balancing.

Before you start

The Workload Balancing virtual appliance is a single pre-installed VM designed to run on a XenServer host. Before importing it, review the prerequisite information and considerations.

Prerequisites

- Workload Balancing 8.3.0 and later are compatible with XenServer 8. We recommend using the XenCenter management console to import the virtual appliance.
- If you perform a rolling pool upgrade from Citrix Hypervisor 8.2 CU1 to XenServer 8, you cannot use Workload Balancing 8.2.2 and earlier with your XenServer 8 pools. Update your Workload Balancing virtual appliance to version 8.3.0 before performing the rolling pool upgrade. You can download the latest version of the Workload Balancing virtual appliance from the XenServer Downloads page.
- If you are currently using an earlier version of the Workload Balancing virtual appliance, you can use the migrate script to migrate your existing data when you upgrade to the latest version. For more information, see Migrate from an existing virtual appliance.
- The Workload Balancing virtual appliance requires a minimum of 2 GB of RAM and 30 GB of disk space to run. By default, the Workload Balancing virtual appliance is assigned 2 vCPUs. This value is sufficient for pools hosting 1000 VMs. You do not usually need to increase it. Only decrease the number of vCPUs assigned to the virtual appliance if you have a small environment. For more information, see Change the Workload Balancing virtual appliance configuration.

Pool requirements

To balance a pool with Workload Balancing, the pool must meet the following requirements:

- All hosts are licensed with a Premium Edition license
- All hosts meet the requirements for live migration:
  - Shared remote storage
  - Similar processor configurations
  - Gigabit Ethernet
- The pool does not contain any vGPU-enabled VMs. Workload Balancing cannot create a capacity plan for VMs that have vGPUs attached.

A single Workload Balancing virtual appliance can manage multiple pools up to a maximum of 100 pools, depending on the virtual appliance's resources (vCPU, memory, disk size). Across these pools, the virtual appliance can manage up to 1000 VMs. However, if a pool has a large number of VMs (for example, more than 400 VMs), we recommend that you use one Workload Balancing virtual appliance just for that pool.

Considerations

Before importing the virtual appliance, note the following information and make the appropriate changes to your environment, as applicable.

- **Communications port**. Before you launch the Workload Balancing Configuration wizard, determine the port over which you want the Workload Balancing virtual appliance to communicate. You are prompted for this port during Workload Balancing Configuration. By default, the Workload Balancing server uses 8012.

  > **Note:**
  >
  > Do not set the Workload Balancing port to port 443. The Workload Balancing virtual appliance cannot accept connections over port 443 (the standard TLS/HTTPS port).

- **Accounts for Workload Balancing**. There are three different accounts that are used when configuring your Workload Balancing virtual appliance and connecting it to XenServer.

  The Workload Balancing Configuration wizard creates the following accounts with a user name and password that you specify:

  - *Workload Balancing account*

    This account is used by the XenServer host to connect to the Workload Balancing server. By default, the user name for this account is `wlbuser`. This user is created on the Workload Balancing virtual appliance during Workload Balancing configuration.

  - *Database account*

    This account is used to access the PostgreSQL database on the Workload Balancing virtual appliance. By default, the user name is `postgres`. You set the password for this account during Workload Balancing configuration.

  When connecting the Workload Balancing virtual appliance to a XenServer pool, you must specify an existing account:

  - *XenServer account*

    This account is used by the Workload Balancing virtual appliance to connect to the XenServer pool and read the RRDs. Ensure that this user account has the permissions to read the XenServer pool, host, and VM RRDs. For example, provide the credentials for a user that has the `pool-admin` or `pool-operator` role.

- **Monitoring across pools**. You can put the Workload Balancing virtual appliance in one pool and monitor a different pool with it. (For example, the Workload Balancing virtual appliance is in Pool A but you are using it to monitor Pool B.)

- **Time synchronization**. The Workload Balancing virtual appliance requires that the time on the physical computer hosting the virtual appliance matches that in use by the monitored pool. There is no way to change the time on the Workload Balancing virtual appliance. We recommend pointing both the physical computer hosting Workload Balancing and the hosts in the pool it is monitoring to the same Network Time (NTP) server.
- **XenServer and Workload Balancing communicate over HTTPS**. Therefore, during Workload Balancing Configuration, Workload Balancing automatically creates a self-signed certificate on your behalf. You can change this certificate to one from a certificate authority or configure XenServer to verify the certificate or both. For information, see the Certificates.
- **Storing historical data and disk space size**. The amount of historical data you can store is based on the following:

  - The size of the virtual disk allocated to Workload Balancing (by default 30 GB)
  - The minimum disk required space, which is 2,048 MB by default and controlled by the `GroomingRequiredMinimumDiskSizeInMB` parameter in the `wlb.conf` file.

  The more historical data Workload Balancing collects, the more accurate and balanced the recommendations are. If you want to store much historical data, you can do one of the following:

  - Archive the data as described in Archive database data
  - Make the virtual disk size assigned to the Workload Balancing virtual appliance larger as described in Extend the virtual appliance disk

  For example, when you want to use the Workload Balancing Pool Audit trail feature and configure the report granularity to medium or above.
- **Load balancing Workload Balancing**. If you want to use your Workload Balancing virtual appliance to manage itself, specify shared remote storage when importing the virtual appliance.

  > **Note:**
  >
  > Workload Balancing cannot perform Start On placement recommendation for the Workload Balancing virtual appliance when you are using Workload Balancing to manage itself. The reason that Workload Balancing cannot make placement recommendations when it is managing itself is because the virtual appliance must be running to perform that function. However, it can balance the Workload Balancing virtual appliance just like it would balance any other VM it is managing.

- **Plan for resource pool sizing**. Workload Balancing requires specific configurations to run successfully in large pools. For more information, see Change the Workload Balancing virtual appliance configuration.

Download the virtual appliance

The Workload Balancing virtual appliance is packaged in an `.xva` format. You can download the virtual appliance from the XenServer Downloads page. When downloading the file, save it to a folder on your local hard drive (typically on the computer where XenCenter is installed).

When the `.xva` download is complete, you can import it into XenCenter as described in Import the Workload Balancing virtual appliance.

Import the Workload Balancing virtual appliance

Use XenCenter to import the Workload Balancing virtual appliance into a pool.

To import the virtual appliance into XenServer:

1. Open XenCenter.
2. Right-click on the pool (or host) into which you want to import the virtual appliance package, and select **Import**.
3. Browse to the `vpx-wlb.xva` package.
4. Select the pool or Home Server where you want to run the Workload Balancing virtual appliance.

   When you select the pool, the VM automatically starts on the most suitable host in that pool.

   Alternatively, if you don't manage the Workload Balancing virtual appliance using Workload Balancing, you can set a Home Server for the Workload Balancing virtual appliance. This setting ensures that the virtual appliance always starts on the same host.
5. Choose a storage repository on which to store the virtual disk for the Workload Balancing virtual appliance. This repository must have a minimum of 30 GB of free space.

   You can choose either local or remote storage. However, if you choose local storage, you cannot manage the virtual appliance with Workload Balancing.
6. Define the virtual interfaces for the Workload Balancing virtual appliance. In this release, Workload Balancing is designed to communicate on a single virtual interface.
7. Choose a network that can access the pool you want Workload Balancing to manage.
8. Leave the **Start VMs after import** check box enabled, and click **Finish** to import the virtual appliance.
9. After you finish importing the Workload Balancing `.xva` file, the Workload Balancing VM appears in the **Resource** pane in XenCenter.

After importing the Workload Balancing virtual appliance, configure the virtual appliance as described in Configure the Workload Balancing virtual appliance.

Configure the Workload Balancing virtual appliance

After you finish importing the Workload Balancing virtual appliance, you must configure it before you can use it to manage your pool. To guide you through the configuration, the Workload Balancing virtual appliance provides you with a configuration wizard in XenCenter. To display it, select the virtual appliance in the **Resource** pane and click the **Console** tab. For all options, press **Enter** to accept the default choice.

1. After importing the Workload Balancing virtual appliance, click the **Console** tab.
2. Enter `yes` to accept the terms of the license agreement. To decline the EULA, enter `no`.

   > **Note:**
   >
   > The Workload Balancing virtual appliance is also subject to the licenses contained in the `/opt/vpx/wlb` directory in the Workload Balancing virtual appliance.

3. Enter and confirm a new root password for the Workload Balancing VM. We recommend selecting a strong password.

   > **Note:**
   >
   > When you enter the password, the console does not display placeholders, such as asterisks, for the characters.

4. Enter the computer name you want to assign to the Workload Balancing virtual appliance.
5. Enter the domain suffix for the virtual appliance.

   For example, if the fully qualified domain name (FQDN) for the virtual appliance is `wlb-vpx-pos-pool.domain4.bedford4.ctx`, enter `domain4.bedford4.ctx`.

   > **Note:**
   >
   > The Workload Balancing virtual appliance does not automatically add its FQDN to your Domain Name System (DNS) server. Therefore, if you want the pool to use an FQDN to connect to Workload Balancing, you must add the FQDN to your DNS server.

6. Enter `y` to use DHCP to obtain the IP address automatically for the Workload Balancing VM. Otherwise, enter `n` and then enter a static IP address, subnet mask, and gateway for the VM.

   > **Note:**
   >
   > Using DHCP is acceptable provided the lease of the IP address does not expire. It is important that the IP address does not change: When it changes, it breaks the connection between XenServer and Workload Balancing.

7. Enter a user name for the Workload Balancing database, or press **Enter** to use the default user name (postgres) of the database account.

   You are creating an account for the Workload Balancing database. The Workload Balancing services use this account to read/write to the Workload Balancing database. Note the user name and password. You might need them if you ever want to administer to the Workload Balancing PostgreSQL database directly (for example, if you wanted to export data).

8. Enter a password for the Workload Balancing database. After pressing **Enter**, messages appear stating that the Configuration wizard is loading database objects.

9. Enter a user name and password for the Workload Balancing Server.

   This action creates the account XenServer uses to connect to Workload Balancing. The default user name is **wlbuser**.

10. Enter the port for the Workload Balancing Server. The Workload Balancing server communicates by using this port.

    By default, the Workload Balancing server uses 8012. The port number cannot be set to 443, which is the default TLS port number.

    > **Note:**
    >
    > If you change the port here, specify that new port number when you connect the pool to Workload Balancing. For example, by specifying the port in the **Connect to WLB Server** dialog.

    Ensure that the port you specify for Workload Balancing is open in any firewalls.

    After you press **Enter**, Workload Balancing continues with the virtual appliance configuration, including creating self-signed certificates.

11. Now, you can also log in to the virtual appliance by entering the VM user name (typically `root`) and the root password you created earlier. However, logging in is only required when you want to run Workload Balancing commands or edit the Workload Balancing configuration file.

After configuring Workload Balancing, connect your pool to the Workload Balancing virtual appliance as described in Connect to the Workload Balancing virtual appliance.

If necessary, you can find the Workload Balancing configuration file in the following location: `/opt/vpx/wlb/wlb.conf`. For more information, see Edit the Workload Balancing configuration file

The Workload Balancing log file is in this location: `/var/log/wlb/LogFile.log`. For more information, see Increase the detail in the Workload Balancing log.

Connect to the Workload Balancing virtual appliance

After configuring Workload Balancing, connect the pools you want managed to the Workload Balancing virtual appliance by using either the CLI or XenCenter.

> **Note:**
>
> A single Workload Balancing virtual appliance can manage multiple pools up to a maximum of 100 pools, depending on the virtual appliance's resources (vCPU, memory, disk size). Across these pools, the virtual appliance can manage up to 1000 VMs. However, if a pool has a large number of VMs (for example, more than 400 VMs), we recommend that you use one Workload Balancing virtual appliance just for that pool.

To connect a pool to your Workload Balancing virtual appliance, you need the following information:

- IP address or FQDN of the Workload Balancing virtual appliance
  - To obtain the IP address for the Workload Balancing virtual appliance:
    1. In XenCenter, go to the Workload Balancing virtual appliance **Console** tab.
    2. Log in as `root` with the root password you created when you imported the appliance.
    3. Run the following command: `ifconfig`.
  - To specify the Workload Balancing FQDN when connecting to the Workload Balancing server, first add its host name and IP address to your DNS server.
- The port number of the Workload Balancing virtual appliance. By default, XenServer connects to Workload Balancing on port 8012.

  Only edit the port number when you have changed it during Workload Balancing Configuration. The port number specified during Workload Balancing Configuration, in any firewall rules, and in the **Connect to WLB Server** dialog must match.
- Credentials for the Workload Balancing account you created during Workload Balancing configuration.

  This account is often known as the Workload Balancing user account. XenServer uses this account to communicate with Workload Balancing. You created this account on the Workload Balancing virtual appliance during Workload Balancing Configuration.
- Credentials for the resource pool (that is, the pool coordinator) you want Workload Balancing to monitor.

  This account is used by the Workload Balancing virtual appliance to connect to the XenServer pool. This account is created on the XenServer pool coordinator and has the `pool-admin` or `pool-operator` role.

When you first connect to Workload Balancing, it uses the default thresholds and settings for balancing workloads. Automatic features, such as Automated Optimization Mode, Power Management, and Automation, are disabled by default.

Working with certificates

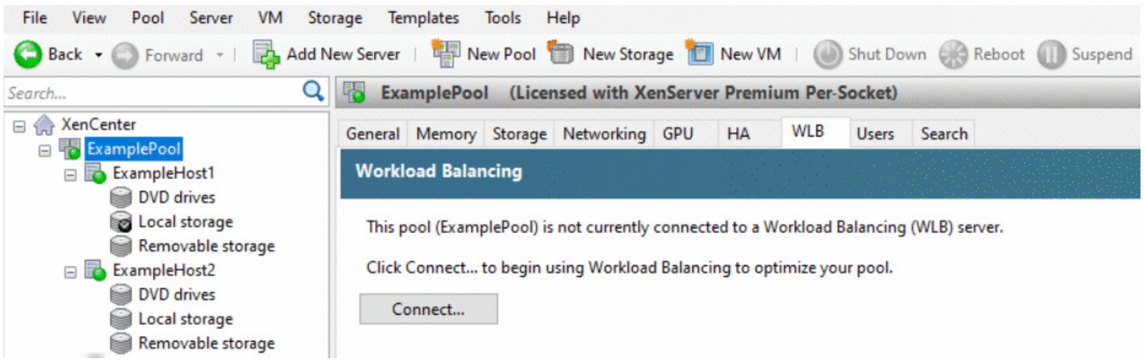
If you want to upload a different (trusted) certificate or configure certificate verification, note the following before connecting your pool to Workload Balancing:

- If you want XenServer to verify the self-signed Workload Balancing certificate, you must use the Workload Balancing IP address to connect to Workload Balancing. The self-signed certificate is issued to Workload Balancing based on its IP address.
- If you want to use a certificate from a certificate authority, it is easier to specify the FQDN when connecting to Workload Balancing. However, you can specify a static IP address in the **Connect to WLB Server** dialog. Use this IP address as the Subject Alternative Name (SAN) in the certificate.

For more information, see Certificates.

To connect your pool to the Workload Balancing virtual appliance

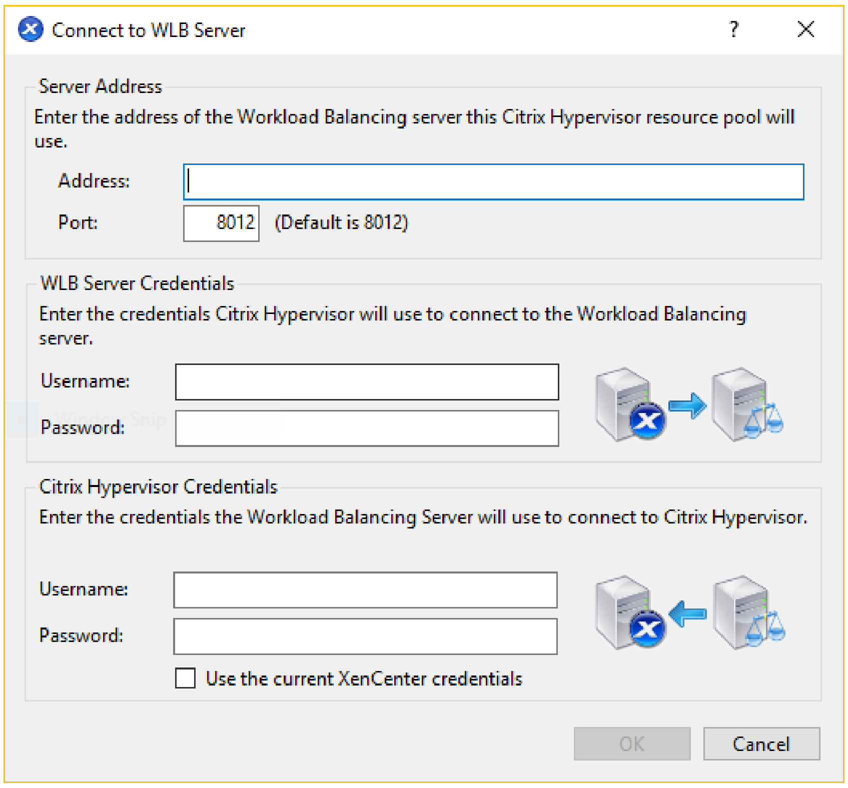
1. In XenCenter, select your resource pool and in its **Properties** pane, click the **WLB** tab. The **WLB** tab displays the **Connect** button.



This pool (ExamplePool) is not currently connected to a Workload Balancing (WLB) server.

Click Connect... to begin using Workload Balancing to optimize your pool.

2. In the **WLB** tab, click **Connect**. The **Connect to WLB Server** dialog box appears.


Connect to WLB Server

Server Address
Enter the address of the Workload Balancing server this Citrix Hypervisor resource pool will use.
Address:
Port: 8012 (Default is 8012)

WLB Server Credentials
Enter the credentials Citrix Hypervisor will use to connect to the Workload Balancing server.
Username:
Password:

Citrix Hypervisor Credentials
Enter the credentials the Workload Balancing Server will use to connect to Citrix Hypervisor.
Username:
Password:
☐ Use the current XenCenter credentials

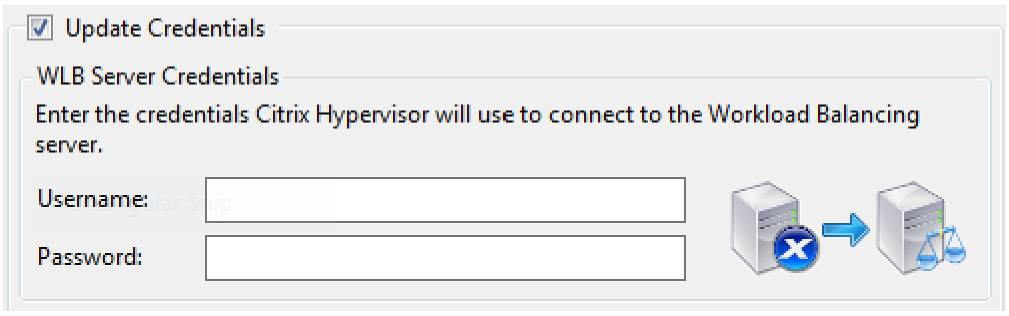
3. In the **Server Address** section, enter the following:

   a) In the **Address** box, type the IP address or FQDN of the Workload Balancing virtual appliance. For example, `WLB-appliance-computername.yourdomain.net`.

   b) (Optional) If you changed the Workload Balancing port during Workload Balancing Configuration, enter the port number in the **Port** box. XenServer uses this port to communicate with Workload Balancing.

      By default, XenServer connects to Workload Balancing on port 8012.

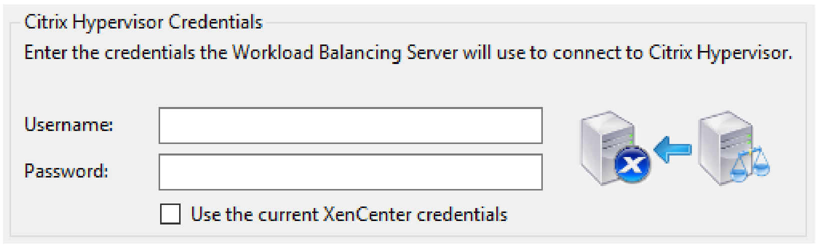
4. In the **WLB Server Credentials** section, enter the user name and password that the pool uses to connect to the Workload Balancing virtual appliance.


☑ Update Credentials

WLB Server Credentials
Enter the credentials Citrix Hypervisor will use to connect to the Workload Balancing server.
Username:
Password:

These credentials must be the account you created during Workload Balancing configuration. By default, the user name for this account is `wlbuser`.

5. In the **Citrix Hypervisor Credentials** section, enter the user name and password for the pool you are configuring. Workload Balancing uses these credentials to connect to the hosts in the pool.

   Citrix Hypervisor Credentials
   Enter the credentials the Workload Balancing Server will use to connect to Citrix Hypervisor.

   Username:
   Password:
   ☐ Use the current XenCenter credentials

   To use the credentials with which you are currently logged into XenServer, select **Use the current XenCenter credentials**. If you have assigned a role to this account using the Access Control feature (RBAC), ensure that the role has sufficient permissions to configure Workload Balancing. For more information, see Workload Balancing Access Control Permissions.

After connecting the pool to the Workload Balancing virtual appliance, Workload Balancing automatically begins monitoring the pool with the default optimization settings. To modify these settings or change the priority given to specific resources, wait at least 60 seconds before proceeding. Or wait until the XenCenter Log shows that discovery is finished.

> **Important:**
>
> After Workload Balancing runs for a time, if you don't receive optimal placement recommendations, evaluate your performance thresholds. This evaluation is described in Understand when Workload Balancing makes recommendations. It is critical to set Workload Balancing to the correct thresholds for your environment or its recommendations might not be appropriate.

Migrate data from an existing virtual appliance

If you are using the Workload Balancing virtual appliance provided with XenServer, you can use the migrate script to migrate your existing data when you upgrade to the latest version (Workload Balancing 8.2.1 or later).

The version of Workload Balancing currently provided with XenServer is 8.3.0. However, Workload Balancing 8.2.0, 8.2.1, and 8.2.2 were previously available with earlier versions of Citrix Hypervisor. You can also use this migration script to migrate from Workload Balancing 8.2.1 or 8.2.2 to Workload Balancing 8.3.0.

To use the migrate script, you must have the following information:

- The root password of the existing Workload Balancing virtual appliance for remote SSH access
- The password of the data base user `postgres` on the existing Workload Balancing virtual appliance

- The password of the data base user `postgres` on the new Workload Balancing virtual appliance

Leave the existing Workload Balancing virtual appliance running on your pool while you complete the migration steps.

1. Follow the steps in the preceding section to import the new Workload Balancing virtual appliance.
2. In the SSH console of the new Workload Balancing virtual appliance, run one of the following commands.

    - For the Workload Balancing 8.2.1 virtual appliance, run:

      ```
      /opt/vpx/wlb/migrate_db.sh 8.2.1 <IP of existing Workload
          Balancing appliance>
      ```

    - For the Workload Balancing 8.2.2 virtual appliance, run:

      ```
      /opt/vpx/wlb/migrate_db.sh 8.2.2 <IP of existing Workload
          Balancing appliance>
      ```

    The command prompts you for password information when required.
3. Connect the XenServer pool with the new Workload Balancing virtual appliance.
4. After you are satisfied with the behavior of this version of the Workload Balancing virtual appliance, you can archive the old version of the virtual appliance.

> **Notes:**
>
> - In the case of a non-recoverable failure, reimport the latest version of the Workload Balancing virtual appliance.
> - Do not disconnect the existing Workload Balancing virtual appliance. Otherwise, the data on the existing virtual appliance is removed.
> - Keep the existing Workload Balancing virtual appliance until you have ensured that the new Workload Balancing virtual appliance is working as required.
> - If necessary, you can roll back this migration by reconnecting the old version of the Workload Balancing virtual appliance to the XenServer pool.

Workload Balancing basic tasks

March 13, 2024

When you first begin using Workload Balancing, there are some basic tasks you use Workload Balancing for regularly:

- Determining the best host on which to run a VM
- Accepting Workload Balancing optimization recommendations
- Running reports about the workloads in your environment

In addition to enabling you to perform these basic tasks, Workload Balancing is a powerful XenServer component that optimizes the workloads in your environment. The features that enable you to optimize your workloads include:

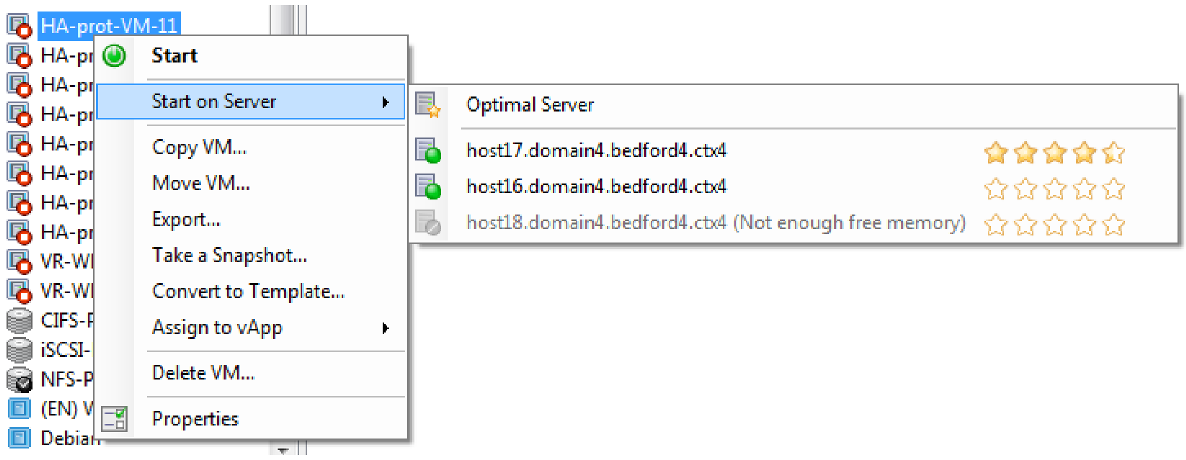
- Host power management
- Scheduling optimization-mode changes
- Running reports
- Fine-tuning the criteria Workload Balancing uses to make optimization recommendations.

For more information about these more complex features, see Administer Workload Balancing.

> **Notes:**
>
> - Workload Balancing is available for XenServer Premium Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.
> - Workload Balancing 8.3.0 is compatible with XenServer 8 and Citrix Hypervisor 8.2 CU1.

Determine the best host on which to run a VM

When you have enabled Workload Balancing and you restart an offline VM, XenCenter recommends the optimal pool members to start the VM on. The recommendations are also known as star ratings since stars are used to indicate the best host.



When Workload Balancing is enabled, XenCenter provides star ratings to indicate the optimal hosts for starting a VM. These ratings are also provided:

- When you want to start the VM when it is powered off
- When you want to start the VM when it is suspended
- When you want to migrate the VM to a different host (Migrate and Maintenance Mode)

When you use these features with Workload Balancing enabled, host recommendations appear as star ratings beside the name of the physical host. Five empty stars indicate the lowest-rated, and thus least optimal, host. If you can't start or migrate a VM to a host, the host name is grayed out in the menu command for a placement feature. The reason it cannot accept the VM appears beside it.

The term *optimal* indicates the physical host best suited to hosting your workload. There are several factors Workload Balancing uses when determining which host is optimal for a workload:

- **The amount of resources available on each host in the pool**. When a pool runs in Maximum Performance mode, Workload Balancing tries to balance the VMs across the hosts so that all VMs have good performance. When a pool runs in Maximum Density mode, Workload Balancing places VMs onto hosts as densely as possible while ensuring the VMs have sufficient resources.
- **The optimization mode in which the pool is running (Maximum Performance or Maximum Density)**. When a pool runs in Maximum Performance mode, Workload Balancing places VMs on hosts with the most resources available of the type the VM requires. When a pool runs in Maximum Density mode, Workload Balancing places VMs on hosts that already have VMs running. This approach ensures that VMs run on as few hosts as possible.
- **The amount and type of resources the VM requires**. After Workload Balancing monitors a VM for a while, it uses the VM metrics to make placement recommendations according to the type of resources the VM requires. For example, Workload Balancing might select a host with less available CPU but more available memory if it is what the VM requires.

In general, Workload Balancing functions more effectively and makes better, less frequent optimization recommendations if you start VMs on the hosts it recommends. To follow the host recommendations, use one of the placement features to select the host with the most stars beside it. Placement recommendations can also be useful in Citrix Virtual Desktops environments.

To start a VM on the optimal host

1. In the Resources pane of XenCenter, select the VM you want to start.
2. From the VM menu, select Start on Server and then select one of the following:
    - **Optimal Server**. The optimal server is the physical host that is best suited to the resource demands of the VM you are starting. Workload Balancing determines the optimal server based on its historical records of performance metrics and your placement strategy. The optimal server is the host with the most stars.

- **One of the servers with star ratings** listed under the Optimal Server command. Five stars indicate the most-recommended (optimal) host and five empty stars indicates the least-recommended host.

> **Tip:**
>
> You can also select Start on Server by right-clicking the VM you want to start in the Resources pane.

To resume a VM on the optimal host

1. In the Resources pane of XenCenter, select the suspended VM you want to resume.
2. From the VM menu, select Resume on Server and then select one of the following:

    - **Optimal Server**. The optimal server is the physical host that is best suited to the resource demands of the VM you are starting. Workload Balancing determines the optimal server based on its historical records of performance metrics and your placement strategy. The optimal server is the host with the most stars.
    - **One of the servers with star ratings** listed under the Optimal Server command. Five stars indicate the most-recommended (optimal) host and five empty stars indicates the least-recommended host.

> **Tip:**
>
> You can also select Resume on Server by right-clicking the suspended VM in the Resources pane.

Accept Workload Balancing optimization recommendations

After Workload Balancing is running for a while, it begins to make recommendations about ways in which you can improve your environment. For example, if your goal is to improve VM density on hosts, at some point, Workload Balancing might recommend that you consolidate VMs on a host. If you aren’t running in automated mode, you can choose either to accept this recommendation and apply it or to ignore it.

> **Important:**
>
> After Workload Balancing runs for a time, if you don’t receive optimal placement recommendations, evaluate your performance thresholds. This evaluation is described in Understand when Workload Balancing makes recommendations. It is critical to set Workload Balancing to the correct thresholds for your environment or its recommendations might not be appropriate.

Optimization recommendations are based on the following criteria:

- Placement strategy you select (that is, the optimization mode).

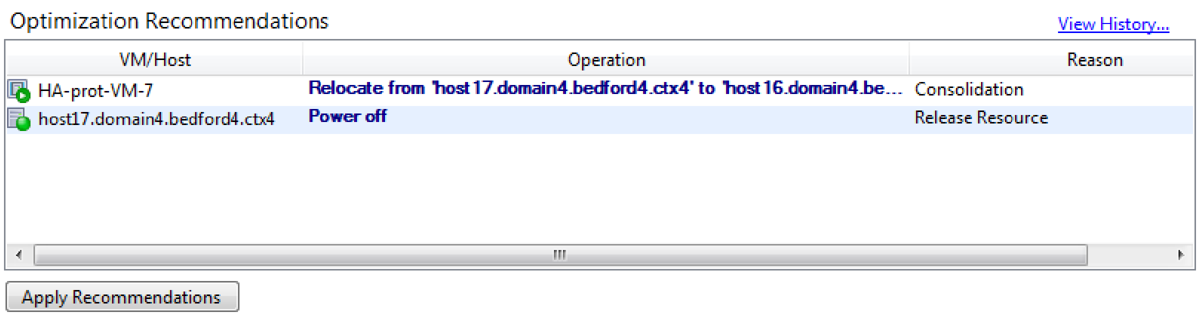
  Find out the optimization mode for a pool by using XenCenter to select the pool. Look in the Configuration section of the WLB tab for the information.

- Performance metrics for resources such as a physical host's CPU, memory, network, and disk utilization.

- The role of the host in the resource pool.

  When making placement recommendations, Workload Balancing considers the pool coordinator for VM placement only if no other host can accept the workload. Likewise, when a pool operates in Maximum Density mode, Workload Balancing considers the pool coordinator last when determining the order to fill hosts with VMs.

Optimization recommendations appear in the **WLB optimization** tab in XenCenter.



VM/Host	Operation	Reason
HA-prot-VM-7	Relocate from 'host17.domain4.bedford4.ctx4' to 'host16.domain4.be...	Consolidation
host17.domain4.bedford4.ctx4	Power off	Release Resource

Optimization recommendations display the following information:

- The name of the VM that Workload Balancing recommends relocating
- The host that the VM currently resides on
- The host Workload Balancing recommends as the new location.

The optimization recommendations also display the reason Workload Balancing recommends moving the VM. For example, the recommendation displays "CPU" to improve CPU utilization. When Workload Balancing power management is enabled, Workload Balancing also displays optimization recommendations for hosts it recommends powering on or off. Specifically, these recommendations are for consolidations.

You can click **Apply Recommendations**, to perform all operations listed in the Optimization Recommendations list.

To accept an optimization recommendation

1. In the Resources pane of XenCenter, select the resource pool for which you want to display recommendations.

2. Click the WLB tab. If there are any recommended optimizations for any VMs on the selected resource pool, they display in the Optimization Recommendations section of the WLB tab.
3. To accept the recommendations, click Apply Recommendations. XenServer begins performing all the operations listed in the Operations column of the Optimization Recommendations section.

   After you click Apply Recommendations, XenCenter automatically displays the Logs tab so you can see the progress of the VM migration.

Understand Workload Balancing recommendations under high availability

If you have Workload Balancing and XenServer High Availability enabled in the same pool, it is helpful to understand how the two features interact. Workload Balancing is designed not to interfere with High Availability. When there is a conflict between a Workload Balancing recommendation and a High Availability setting, the **High Availability** setting always takes precedence. In practice, this precedence means that:

- If attempting to start a VM on a host violates the High Availability plan, Workload Balancing doesn't give you star ratings.
- Workload Balancing does not automatically power off any hosts beyond the number specified in the Failures allowed box in the **Configure HA** dialog.
  - However, Workload Balancing might still make recommendations to power off more hosts than the number of host failures to tolerate. (For example, Workload Balancing still recommends that you power off two hosts when High Availability is only configured to tolerate one host failure.) However, when you attempt to apply the recommendation, XenCenter might display an error message stating that High Availability is no longer guaranteed.
  - When Workload Balancing runs in automated mode and has power management enabled, recommendations that exceed the number of tolerated host failures are ignored. In this situation, the Workload Balancing log shows a message that power-management recommendation wasn't applied because High Availability is enabled.

Generate workload reports

Workload Balancing captures performance data and can use this data to generate reports, known as Workload Reports, about your virtualized environment, including reports about hosts and VMs. The Workload Balancing reports can help you perform capacity planning, determine virtual server health, and evaluate how effective your configured threshold levels are.

You can use the Pool Health report to evaluate how effective your optimization thresholds are. While Workload Balancing provides default threshold settings, you might need to adjust these defaults for them to provide value in your environment. If you do not have the optimization thresholds adjusted to the correct level for your environment, Workload Balancing recommendations might not be appropriate for your environment.

To run reports, you do not need to configure for Workload Balancing to make placement recommendations or move VMs. However, you must configure the Workload Balancing component. Ideally, you must set critical thresholds to values that reflect the point at which the performance of the hosts in your pool degrades. Ideally, the pool has been running Workload Balancing for a couple of hours or long enough to generate the data to display in the reports.

Workload Balancing lets you generate reports on three types of objects: physical hosts, resource pools, and VMs. At a high level, Workload Balancing provides two types of reports:

- Historical reports that display information by date
- "Roll up"style reports, which provide a summarizing overview of an area
- Reports for auditing purposes, so you can determine, for example, the number of times a VM moved
- Chargeback report that shows VM usage and can help you measure and assign costs

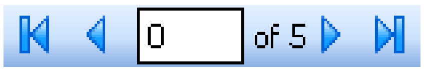
Generate a Workload Balancing report

1. In XenCenter, from the **Pool** menu, select **View Workload Reports**.

   You can also display the **Workload Reports** screen from the **WLB** tab by clicking the **Reports** button.
2. From the **Workload Reports** screen, select a report from the **Reports** pane.
3. Select the **Start Date** and the **End Date** for the reporting period. Depending on the report you select, you might be required to specify a host in the **Host** list.
4. Click **Run Report**. The report displays in the report window. For information about the meaning of the reports, see Workload Balancing report glossary.

Navigate in a Workload Balancing report

After generating a report, you can use the toolbar buttons in the report to navigate and perform certain tasks. To display the name of a toolbar button, pause your mouse over toolbar icon.

Toolbar buttons	Description
	Document Map enables you to display a document map that helps you nav
	**Page Forward/Back** enables you to move one page ahead or back in the rep
	Back to Parent Report enables you to return to the parent report when wor
	Stop Rendering cancels the report generation.
	Print enables you to print a report and specify general printing options. The
	Print Layout enables you to display a preview of the report before you print
	Page Setup enables you to specify printing options such as the paper size, p
	Export enables you to export the report as an Acrobat (.PDF) file or as an Exc
	Find enables you to search for a word in a report, such as the name of a VM.

Export a Workload Balancing report

You can export a report in either Microsoft Excel or Adobe Acrobat (PDF) formats.

1. After generating the report, click the following Export button:
2. Select one of the following items from the Export button menu:

    - Excel
    - Acrobat (PDF) file

> **Note:**
>
> Depending on the export format you select, the report contains different amounts of data. Reports exported to Excel include all the data available for reports, including "drilldown"data. Reports exported to PDF and displayed in XenCenter only contain the data that you selected when you generated the report.

Workload Balancing report glossary

This section provides information about the following Workload Balancing reports:

- Chargeback Utilization Analysis
- Host Health History
- Pool Optimization Performance History

- Pool Audit Trail
- Pool Health
- Pool Health History
- Pool Optimization History
- Virtual Machine Motion History
- Virtual Machine Performance History

Chargeback Utilization Analysis

You can use the Chargeback Utilization Analysis report ("chargeback report") to determine how much of a resource a specific department in your organization used. Specifically, the report shows information about all the VMs in your pool, including their availability and resource utilization. Since this report shows VM up time, it can help you demonstrate Service Level Agreements compliance and availability.

The chargeback report can help you implement a simple chargeback solution and facilitate billing. To bill customers for a specific resource, generate the report, save it as Excel, and edit the spreadsheet to include your price per unit. Alternatively, you can import the Excel data into your billing system.

If you want to bill internal or external customers for VM usage, consider incorporating department or customer names in your VM naming conventions. This practice makes reading chargeback reports easier.

The resource reporting in the chargeback report is, sometimes, based on the allocation of physical resources to individual VMs.

The average memory data in this report is based on the amount of memory currently allocated to the VM. XenServer enables you to have a fixed memory allocation or an automatically adjusting memory allocation (Dynamic Memory Control).

The chargeback report contains the following columns of data:

- **VM Name**. The name of the VM to which the data in the columns in that row applies.
- **VM Uptime**. The number of minutes the VM was powered on (or, more specifically, appears with a green icon beside it in XenCenter).
- **vCPU Allocation**. The number of virtual CPUs configured on the VM. Each virtual CPU receives an equal share of the physical CPUs on the host. For example, consider the case where you configured eight virtual CPUs on a host that contains two physical CPUs. If the **vCPU Allocation** column has "1"in it, this value is equal to 2/16 of the total processing power on the host.
- **Minimum CPU Usage (%)**. The lowest recorded value for virtual CPU utilization in the reporting period. This value is expressed as a percentage of the VM's vCPU capacity. The capacity is based on the number of vCPUs allocated to the VM. For example, if you allocated one vCPU to a VM,

Minimum CPU Usage represents the lowest percentage of vCPU usage that is recorded. If you allocated two vCPUs to the VM, the value is the lowest usage of the combined capacity of both vCPUs as a percentage.

Ultimately, the percentage of CPU usage represents the lowest recorded workload that virtual CPU handled. For example, if you allocate one vCPU to a VM and the pCPU on the host is 2.4 GHz, 0.3 GHz is allocated to the VM. If the **Minimum CPU Usage** for the VM was 20%, the VM's lowest usage of the physical host's CPU during the reporting period was 60 MHz.

- **Maximum CPU Usage (%)**. The highest percentage of the VM's virtual CPU capacity that the VM consumed during the reporting period. The CPU capacity consumed is a percentage of the virtual CPU capacity you allocated to the VM. For example, if you allocated one vCPU to the VM, the Maximum CPU Usage represents the highest recorded percentage of vCPU usage during the time reported. If you allocated two virtual CPUs to the VM, the value in this column represents the highest utilization from the combined capacity of both virtual CPUs.
- **Average CPU Usage (%)**. The average amount, expressed as a percentage, of the VM's virtual CPU capacity that was in use during the reporting period. The CPU capacity is the virtual CPU capacity you allocated to the VM. If you allocated two virtual CPUs to the VM, the value in this column represents the average utilization from the combined capacity of both virtual CPUs.
- **Total Storage Allocation (GB)**. The amount of disk space that is currently allocated to the VM at the time the report was run. Frequently, unless you modified it, this disk space is the amount of disk space you allocated to the VM when you created it.
- **Virtual NIC Allocation**. The number of virtual interfaces (VIFs) allocated to the VM.
- **Current Minimum Dynamic Memory (MB)**.
  - **Fixed memory allocation**. If you assigned a VM a fixed amount of memory (for example, 1,024 MB), the same amount of memory appears in the following columns: Current Minimum Dynamic Memory (MB), Current Maximum Dynamic Memory (MB), Current Assigned Memory (MB), and Average Assigned Memory (MB).
  - **Dynamic memory allocation**. If you configured XenServer to use Dynamic Memory Control, the minimum amount of memory specified in the range appears in this column. If the range has 1,024 MB as minimum memory and 2,048 MB as maximum memory, the **Current Minimum Dynamic Memory (MB)** column displays 1,024 MB.
- **Current Maximum Dynamic Memory (MB)**.
  - **Dynamic memory allocation**. If XenServer adjusts a VM's memory automatically based on a range, the maximum amount of memory specified in the range appears in this column. For example, if the memory range values are 1,024 MB minimum and 2,048 MB maximum, 2,048 MB appears in the **Current Maximum Dynamic Memory (MB)** column.

  - **Fixed memory allocation**. If you assign a VM a fixed amount of memory (for example, 1,024 MB), the same amount of memory appears in the following columns: Current Minimum Dynamic Memory (MB), Current Maximum Dynamic Memory (MB), Current Assigned Memory (MB), and Average Assigned Memory (MB).

- **Current Assigned Memory (MB)**.
  - **Dynamic memory allocation**. When Dynamic Memory Control is configured, this value indicates the amount of memory XenServer allocates to the VM when the report runs.
  - **Fixed memory allocation**. If you assign a VM a fixed amount of memory (for example, 1,024 MB), the same amount of memory appears in the following columns: Current Minimum Dynamic Memory (MB), Current Maximum Dynamic Memory (MB), Current Assigned Memory (MB), and Average Assigned Memory (MB).

  > **Note:**
  >
  > If you change the VM's memory allocation immediately before running this report, the value reflected in this column reflects the new memory allocation you configured.

- **Average Assigned Memory (MB)**.
  - **Dynamic memory allocation**. When Dynamic Memory Control is configured, this value indicates the average amount of memory XenServer allocated to the VM over the reporting period.
  - **Fixed memory allocation**. If you assign a VM a fixed amount of memory (for example, 1,024 MB), the same amount of memory appears in the following columns: Current Minimum Dynamic Memory (MB), Current Maximum Dynamic Memory (MB), Current Assigned Memory (MB), and Average Assigned Memory (MB).

  > **Note:**
  >
  > If you change the VM's memory allocation immediately before running this report, the value in this column might not change from what was previously displayed. The value in this column reflects the average over the time period.

- **Average Network Reads (BPS)**. The average amount of data (in bits per second) the VM received during the reporting period.
- **Average Network Writes (BPS)**. The average amount of data (in bits per second) the VM sent during the reporting period.
- **Average Network Usage (BPS)**. The combined total (in bits per second) of the Average Network Reads and Average Network Writes. If a VM sends, on average, 1,027 bps and receives, on average, 23,831 bps during the reporting period, the Average Network Usage is the combined total of these values: 24,858 bps.

- **Total Network Usage (BPS)**. The total of all network read and write transactions in bits per second over the reporting period.

Host Health History

This report displays the performance of resources (CPU, memory, network reads, and network writes) on specific host in relation to threshold values.

The colored lines (red, green, yellow) represent your threshold values. You can use this report with the Pool Health report for a host to determine how the host's performance might affect overall pool health. When you are editing the performance thresholds, you can use this report for insight into host performance.

You can display resource utilization as a daily or hourly average. The hourly average lets you see the busiest hours of the day, averaged, for the time period.

To view report data which is grouped by hour, under **Host Health History** expand **Click to view report data grouped by house for the time period**.

Workload Balancing displays the average for each hour for the time period you set. The data point is based on a utilization average for that hour for all days in the time period. For example, in a report for May 1, 2009, to May 15, 2009, the Average CPU Usage data point represents the resource utilization of all 15 days at 12:00 hours. This information is combined as an average. If CPU utilization was 82% at 12PM on May 1, 88% at 12PM on May 2, and 75% on all other days, the average displayed for 12PM is 76.3%.

> **Note:**
>
> Workload Balancing smoothes spikes and peaks so data does not appear artificially high.

Pool Optimization Performance History

The optimization performance report displays optimization events against that pool's average resource usage. These events are instances when you optimized a resource pool. Specifically, it displays resource usage for CPU, memory, network reads, and network writes.

The dotted line represents the average usage across the pool over the period of days you select. A blue bar indicates the day on which you optimized the pool.

This report can help you determine if Workload Balancing is working successfully in your environment. You can use this report to see what led up to optimization events (that is, the resource usage before Workload Balancing recommended optimizing).

This report displays average resource usage for the day. It does not display the peak utilization, such as when the system is stressed. You can also use this report to see how a resource pool is performing when Workload Balancing is not making optimization recommendations.

In general, resource usage declines or stays steady after an optimization event. If you do not see improved resource usage after optimization, consider readjusting threshold values. Also, consider whether the resource pool has too many VMs and whether you added or removed new VMs during the period that you specified.

Pool Audit Trail

This report displays the contents of the XenServer Audit Log. The Audit Log is a XenServer feature designed to log attempts to perform unauthorized actions and select authorized actions. These actions include:

- Import and export
- Host and pool backups
- Guest and host console access.

The report gives more meaningful information when you give XenServer administrators their own user accounts with distinct roles assigned to them by using the RBAC feature.

> **Important:**
>
> To run the audit log report, you must enable the Audit Logging feature. By default, Audit Log is always enabled in the Workload Balancing virtual appliance.

The enhanced Pool Audit Trail feature allows you to specify the granularity of the audit log report. You can also search and filter the audit trail logs by specific users, objects, and by time. The Pool Audit Trail Granularity is set to Minimum by default. This option captures limited amount of data for specific users and object types. You can modify the setting at any time based on the level of detail you require in your report. For example, set the granularity to Medium for a user-friendly report of the audit log. If you require a detailed report, set the option to Maximum.

The Pool Audit Trail report contains the following information:

- Time. The time XenServer recorded the user's action.
- User Name. The name of the person who created the session in which the action was performed. Sometimes, this value can be the User ID
- Event Object. The object that was the subject of the action (for example, a VM).
- Event Action. The action that occurred. For definitions of these actions, see Audit Log Event Names.

- Access. Whether the user had permission to perform the action.
- Object Name. The name of the object (for example, the name of the VM).
- Object UUID. The UUID of the object (for example, the UUID of the VM).
- Succeeded. This information provides the status of the action (that is, whether it was successful).

Audit Log event names The Audit Log report logs XenServer events, event objects and actions, including import/export, host and pool backups, and guest and host console access. The following table defines some of the typical events that appear frequently in the XenServer Audit Log and Pool Audit Trail report. The table also specifies the granularity of these events.

In the Pool Audit Trail report, the events listed in the `Event Action` column apply to a pool, VM, or host. To determine what the events apply to, see the `Event Object` and `Object Name` columns in the report. For more event definitions, see the events section of the XenServer Management API.

Pool Audit Trail Granularity	Event Action	User Action
Minimum	`pool.join`	Instructed the host to join a new pool
Minimum	`pool.join_force`	Instructed (forced) the host to join a pool
Medium	`SR.destroy`	Destroyed the storage repository
Medium	`SR.create`	Created a storage repository
Medium	`VDI.snapshot`	Took a read-only snapshot of the VDI, returning a reference to the snapshot
Medium	`VDI.clone`	Took an exact copy of the VDI, returning a reference to the new disk
Medium	`VIF.plug`	Hot-plugged the specified VIF, dynamically attaching it to the running VM
Medium	`VIF.unplug`	Hot-unplugged the specified VIF, dynamically detaching it from the running VM

Pool Audit Trail Granularity	Event Action	User Action
Maximum	`auth.get_subject_identifier`	Queried the external directory service to obtain the subject identifier as a string from the human-readable subject name
Maximum	`task.cancel`	Requested that a task is canceled
Maximum	`VBD.insert`	Inserted new media into the device
Maximum	`VIF.get_by_uuid`	Obtained a reference to the VIF instance with the specified UUID
Maximum	`VDI.get_sharable`	Obtained the shareable field of the given VDI
Maximum	`SR.get_all`	Returns a list of all SRs known to the system
Maximum	`pool.create_new_blob`	Created a placeholder for a named binary piece of data that is associated with this pool
Maximum	`host.send_debug_keys`	Injected the given string as debugging keys into Xen
Maximum	`VM.get_boot_record`	Returned a record describing the VMs dynamic state, initialized when the VM boots, and updated to reflect runtime configuration changes, for example, CPU hotplug

Pool Health

The Pool Health report displays the percentage of time a resource pool and its hosts spent in four different threshold ranges: Critical, High, Medium, and Low. You can use the Pool Health report to evaluate the effectiveness of your performance thresholds.

A few points about interpreting this report:

- Resource utilization in the Average Medium Threshold (blue) is the optimum resource utilization regardless of the placement strategy you selected. Likewise, the blue section on the pie chart indicates the amount of time that host used resources optimally.

- Resource utilization in the Average Low Threshold Percent (green) is not necessarily positive. Whether Low resource utilization is positive depends on your placement strategy. If your placement strategy is Maximum Density and resource usage is green, Workload Balancing might not be fitting the maximum number of VMs on that host or pool. If so, adjust your performance threshold values until most of your resource utilization falls into the Average Medium (blue) threshold range.
- Resource utilization in the Average Critical Threshold Percent (red) indicates the amount of time average resource utilization met or exceeded the Critical threshold value.

If you double-click on a pie chart for a host's resource usage, XenCenter displays the Host Health History report for that resource on that host. Clicking **Back to Parent Report** on the toolbar returns you to the Pool Health history report.

If you find that most of your report results are not in the Average Medium Threshold range, adjust the Critical threshold for this pool. While Workload Balancing provides default threshold settings, these defaults are not effective in all environments. If you do not have the thresholds adjusted to the correct level for your environment, the Workload Balancing optimization and placement recommendations might not be appropriate. For more information, see Change the critical thresholds.

Pool Health History

This report provides a line graph of resource utilization on all physical hosts in a pool over time. It lets you see the trend of resource utilization—if it tends to be increasing in relation to your thresholds (Critical, High, Medium, and Low). You can evaluate the effectiveness of your performance thresholds by monitoring trends of the data points in this report.

Workload Balancing extrapolates the threshold ranges from the values you set for the Critical thresholds when you connected the pool to Workload Balancing. Although similar to the Pool Health report, the Pool Health History report displays the average utilization for a resource on a specific date. Instead of the amount of time overall the resource spent in a threshold.

Except for the Average Free Memory graph, the data points never average above the Critical threshold line (red). For the Average Free Memory graph, the data points never average *below* the Critical threshold line (which is at the bottom of the graph). Because this graph displays *free* memory, the Critical threshold is a low value, unlike the other resources.

A few points about interpreting this report:

- When the Average Usage line in the chart approaches the Average Medium Threshold (blue) line, it indicates the pool's resource utilization is optimum. This indication is regardless of the placement strategy configured.

- Resource utilization approaching the Average Low Threshold (green) is not necessarily positive. Whether Low resource utilization is positive depends on your placement strategy. In the case where:
  - Your placement strategy is Maximum Density
  - Most days the Average Usage line is at or below the green line
    Workload Balancing might not be placing VMs as densely as possible on that pool. If so, adjust the pool's Critical threshold values until most of its resource utilization falls into the Average Medium (blue) threshold range.
- When the Average Usage line intersects with the Average Critical Threshold (red), it indicates when the average resource utilization met or exceeded the Critical threshold value for that resource.

If data points in your graphs aren't in the Average Medium Threshold range, but the performance is satisfactory, you can adjust the Critical threshold for this pool. For more information, see Change the critical thresholds.

Pool Optimization History

The Pool Optimization History report provides chronological visibility into Workload Balancing optimization activity.

Optimization activity is summarized graphically and in a table. Drilling into a date field within the table displays detailed information for each pool optimization performed for that day.

This report lets you see the following information:

- VM Name: The name of the VM that Workload Balancing optimized.
- Reason: The reason for the optimization.
- Method: Whether the optimization was successful.
- From Host: The physical host where the VM was originally hosted.
- To Host: The physical host where the VM was migrated.
- Time: The time when the optimization occurred.

> **Tip:**
>
> You can also generate a Pool Optimization History report from the WLB tab by clicking the View History link.

Virtual Machine Motion History

This line graph displays the number of times VMs migrated on a resource pool over a period. It indicates if a migration resulted from an optimization recommendation and to which host the VM moved. This report also indicates the reason for the optimization. You can use this report to audit the number of migrations on a pool.

Some points about interpreting this report:

- The numbers on the left side of the chart correspond with the number of migrations possible. This value is based on how many VMs are in a resource pool.
- You can look at details of the migrations on a specific date by expanding the + sign in the Date section of the report.

Virtual Machine Performance History

This report displays performance data for each VM on a specific host for a time period you specify. Workload Balancing bases the performance data on the amount of virtual resources allocated for the VM. For example, if Average CPU Usage for your VM is 67%, your VM uses, on average, 67% of its vCPU for the specified period.

The initial view of the report displays an average value for resource utilization over the period you specified.

Expanding the + sign displays line graphs for individual resources. You can use these graphs to see trends in resource utilization over time.

This report displays data for CPU Usage, Free Memory, Network Reads/Writes, and Disk Reads/Writes.

Configure Workload Balancing behavior

March 13, 2024

After connecting to the Workload Balancing virtual appliance, you can edit the settings Workload Balancing uses to calculate placement and recommendations. Workload Balancing settings apply collectively to all VMs and hosts in the pool.

Placement and optimization settings that you can modify include the following:

- Changing the placement strategy
- Configuring automatic optimizations and power management

- Editing performance thresholds and metric weightings
- Excluding hosts.

Provided the network and disk thresholds align with the hardware in your environment, consider using most of the defaults in Workload Balancing initially. After Workload Balancing is enabled for a while, we recommend evaluating your performance thresholds and determining whether to edit them. For example, consider the following cases:

- Getting recommendations when they are not yet required. If so, try adjusting the thresholds until Workload Balancing begins providing suitable recommendations.
- Not getting recommendations when you expect to receive them. For example, if your network has insufficient bandwidth and you do not receive recommendations, you might have to tweak your settings. If so, try lowering the network critical thresholds until Workload Balancing begins providing recommendations.

Before you edit your thresholds, you can generate a Pool Health report and the Pool Health History report for each physical host in the pool. For more information, see Generate workload reports.

> **Notes:**
>
> - Workload Balancing is available for XenServer Premium Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.
> - Workload Balancing 8.3.0 is compatible with XenServer 8 and Citrix Hypervisor 8.2 CU1.

This article assumes that you already connected your pool to a Workload Balancing virtual appliance. For information about downloading, importing, configuring, and connecting to a Workload Balancing virtual appliance, see Get started.

Adjust the optimization mode

Workload Balancing makes recommendations to rebalance, or optimize, the VM workload in your environment based on a strategy for placement you select. The placement strategy is known as the optimization mode.

You can choose from the following optimization modes:

- **Maximize Performance** (default)

  Workload Balancing attempts to spread workload evenly across all physical hosts in a resource pool. The goal is to minimize CPU, memory, and network pressure for all hosts. When Maximize Performance is your placement strategy, Workload Balancing recommends optimization when a host reaches the High threshold.

- **Maximize Density**

  Workload Balancing attempts to minimize the number of physical hosts that must be online by consolidating the active VMs.

  When you select Maximize Density as your placement strategy, you can specify parameters similar to the ones in Maximize Performance. However, Workload Balancing uses these parameters to determine how it can pack VMs onto a host. If Maximize Density is your placement strategy, Workload Balancing recommends consolidation optimizations when a VM reaches the Low threshold.

Workload Balancing also lets you apply these optimization modes always, *fixed*, or switch between modes for specified time periods, *scheduled*:

Fixed optimization modes

Fixed optimization modes set Workload Balancing to have a specific optimization behavior always. This behavior can be either to try to create the best performance or to create the highest density.

To set a fixed optimization mode, complete the following steps:

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, click **Optimization Mode**.
5. In the **Fixed** section of the **Optimization Mode** page, select one of these optimization modes:

   - Maximize Performance (default). Attempts to spread workload evenly across all physical hosts in a resource pool. The goal is to minimize CPU, memory, and network pressure for all hosts.
   - Maximize Density. Attempts to fit as many VMs as possible onto a physical host. The goal is to minimize the number of physical hosts that must be online.

Scheduled optimization modes

Scheduled optimization modes let you schedule for Workload Balancing to apply different optimization modes depending on the time of day. For example, you might want to configure Workload Balancing to optimize for performance during the day when you have users connected. To save energy, you can then specify for Workload Balancing to optimize for Maximum Density at night.

When you configure scheduled optimization modes, Workload Balancing automatically changes to the optimization mode at the beginning of the time period you specified. You can configure Everyday, Weekdays, Weekends, or individual days. For the hour, you select a time of day.

To set a schedule for your optimization modes, complete the following steps:

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, click **Optimization Mode**.
5. In the **Optimization Mode** pane, select **Scheduled**. The **Scheduled** section becomes available.
6. Click **Add New**.
7. In the **Change to** box, select one of the following modes:
    - Maximize Performance. Attempts to spread workload evenly across all physical hosts in a resource pool. The goal is to minimize CPU, memory, and network pressure for all hosts.
    - Maximize Density. Attempts to fit as many VMs as possible onto a physical host. The goal is to minimize the number of physical hosts that must be online.
8. Select the day of the week and the time when you want Workload Balancing to begin operating in this mode.
9. Repeat the preceding steps to create more scheduled mode tasks until you have the number you need. If you only schedule one task, Workload Balancing switches to that mode as scheduled, but then it never switches back.
10. Click **OK**.

To change your schedule settings, complete the following steps.

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, click **Optimization Mode**.
5. Select the task that you want to delete or disable from the **Scheduled Mode Changes** list.
6. Do one of the following:
    - **Delete the task permanently**: Click the **Delete** button.
    - **Stop the task from running temporarily**: Right-click the task and click **Disable**.

> **Tips:**
>
> - You can also disable or enable tasks by selecting the task, clicking **Edit**, and selecting the **Enable Task** check box in the **Optimization Mode Scheduler** dialog.
> - To re-enable a task, right-click the task in the **Scheduled Mode Changes** list and click **Enable**.

- **Edit the task**: Double-click the task that you want to edit. In the **Change to** box, select a different mode or make other changes as desired.

> **Note:**
>
> Clicking Cancel, before clicking OK, undoes any changes you made in the Optimization tab, including deleting a task.

Optimize and manage power automatically

You can configure Workload Balancing to apply recommendations automatically and turn hosts on or off automatically. To power down hosts automatically (for example, during low-usage periods), you must configure Workload Balancing to apply recommendations automatically and enable power management. Both power management and automation are described in the sections that follow.

Apply recommendations automatically

Workload Balancing lets you configure for it to apply recommendations on your behalf and perform the optimization actions it recommends automatically. You can use this feature, which is known as automatic optimization acceptance, to apply any recommendations automatically, including ones to improve performance or power down hosts. However, to power down hosts as VMs usage drops, you must configure automation, power management, and Maximum Density mode.

By default, Workload Balancing does not apply recommendations automatically. If you want Workload Balancing to apply recommendations automatically, enable automation. If you do not, you must apply recommendations manually by clicking **Apply Recommendations**.

Workload Balancing does not automatically apply recommendations to hosts or VMs when the recommendations conflict with HA settings. If a pool becomes overcommitted by applying Workload Balancing optimization recommendations, XenCenter prompts you whether you want to continue applying the recommendation. When automation is enabled, Workload Balancing does not apply any power-management recommendations that exceed the number of host failures to tolerate in the HA plan.

When Workload Balancing is running with the automation feature enabled, this behavior is sometimes called running in automated mode.

It is possible to tune how Workload Balancing applies recommendations in automated mode. For information, see Set conservative or aggressive automated recommendations.

To apply optimization recommendations automatically

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, click **Automation**.
5. Select one or more of the following check boxes:
    - **Automatically apply Optimization recommendations**. When you select this option, you do not need to accept optimization recommendations manually. Workload Balancing automatically accepts the optimization and placement recommendations that it makes.
    - **Automatically apply Power Management recommendations**. The behavior of this option varies according to the optimization mode of the pool:
        - Maximum Performance mode: When **Automatically apply Power Management recommendations** is enabled, Workload Balancing automatically powers on hosts when doing so improves host performance.
        - Maximum Density mode: When **Automatically apply Power Management recommendations** is enabled, Workload Balancing automatically powers off hosts when resource utilization drops below the Low threshold. That is, Workload Balancing powers off hosts automatically during low usage periods.
6. (Optional.) Fine-tune optimization recommendations by clicking **Advanced** in the left pane of the **Settings** dialog and doing one or more of the following actions:
    - Specifying the number of times Workload Balancing must make an optimization recommendation before the recommendation is applied automatically. The default is three times, which means the recommendation is applied on the third time it is made.
    - Selecting the lowest level of optimization recommendation that you want Workload Balancing to apply automatically. The default is High.
    - Changing the aggressiveness with which Workload Balancing applies its optimization recommendations.

    You might also want to specify the number of minutes Workload Balancing has to wait before applying an optimization recommendation to a recently moved VM.

    All of these settings are explained in more depth in Set conservative or aggressive automated recommendations.

7. (optional) If you want to configure power management, click **Automation/Power Management**

   a) In the **Power Management** section, select the hosts that you want Workload Balancing to recommend powering on and off.

      > **Note:**
      >
      > Selecting hosts for power management recommendations without selecting **Automatically apply Power Management recommendations** causes Workload Balancing to suggest power-management recommendations but not apply them automatically for you.

      If none of the hosts in the resource pool support remote power management, Workload Balancing displays the message, "No hosts support Power Management."

   b) Click **OK**.

8. To finish configuring the automation, click **OK**.

Enable Workload Balancing power management

The term power management means the ability to the turn the power on or off for physical hosts. In a Workload Balancing context, this term means powering hosts in a pool on or off based on the total workload of the pool.

Configuring Workload Balancing power management on a host requires that:

- The hardware for the host has remote power on/off capabilities.
- The Host Power On feature is configured for the host. To configure the Host Power On feature for the host, see Configure Host Power On feature.
- The host has been explicitly selected as a host to participate in Workload Balancing power management.

In addition, if you want Workload Balancing to power off hosts automatically, configure Workload Balancing to do the following actions:

- Apply recommendations automatically
- Apply power management recommendations automatically

When a host is set to participate in power management, Workload Balancing makes power-on and power-off recommendations as needed.

If you run in Maximum Density mode:

- When Workload Balancing detects unused resources in a pool, it recommends powering off hosts until it eliminates all excess capacity.
- If there isn't enough host capacity in the pool to shut down hosts, Workload Balancing recommends leaving the hosts on until the pool workload decreases enough.
- When you configure Workload Balancing to power off extra hosts automatically, it applies these recommendations automatically and, so, behaves in the same way.

If you run in Maximum Performance mode:

- If you configure Workload Balancing to power on hosts automatically, Workload Balancing powers on hosts when resource utilization on a host exceeds the High threshold.
- Workload Balancing never powers off hosts after it has powered them on.

If you turn on the option to apply power management recommendations automatically, you do so at the pool level. However, you can specify which hosts from the pool you want to participate in power management.

Configure Host Power On feature To configure the Host Power On feature for your host, follow these steps:

1. In XenCenter, select your host and click **Properties**.
2. In the left pane, click **Power On**.
3. For the **Power On mode**, select **Dell Remote Access Controller (DRAC)**.
4. For the **Configuration options**, enter your server's DRAC IP address. This is the IP address of the BMC management port. For more information, see DRAC Card How To Guide [PDF].
5. After the Dell Remote Access Controller (DRAC) is configured, select your pool.
6. In the **Properties** pane of the pool, click the **WLB** tab.
7. In the **WLB** tab, click **Settings**.
8. In the left pane, click **Automation**.
9. For **Automation**, select the following check boxes:
    - **Automatically apply Optimization recommendations**. When you select this option, you do not need to accept optimization recommendations manually. Workload Balancing automatically accepts the optimization and placement recommendations that it makes.
    - **Automatically apply Power Management recommendations**. The behavior of this option varies according to the optimization mode of the pool:

- Maximum Performance mode: When **Automatically apply Power Management recommendations** is enabled, Workload Balancing automatically powers on hosts when doing so improves host performance.
- Maximum Density mode: When **Automatically apply Power Management recommendations** is enabled, Workload Balancing automatically powers off hosts when resource utilization drops below the Low threshold. That is, Workload Balancing powers off hosts automatically during low usage periods.

10. For **Power Management**, select the name of the **Host Server** that you're currently configuring.

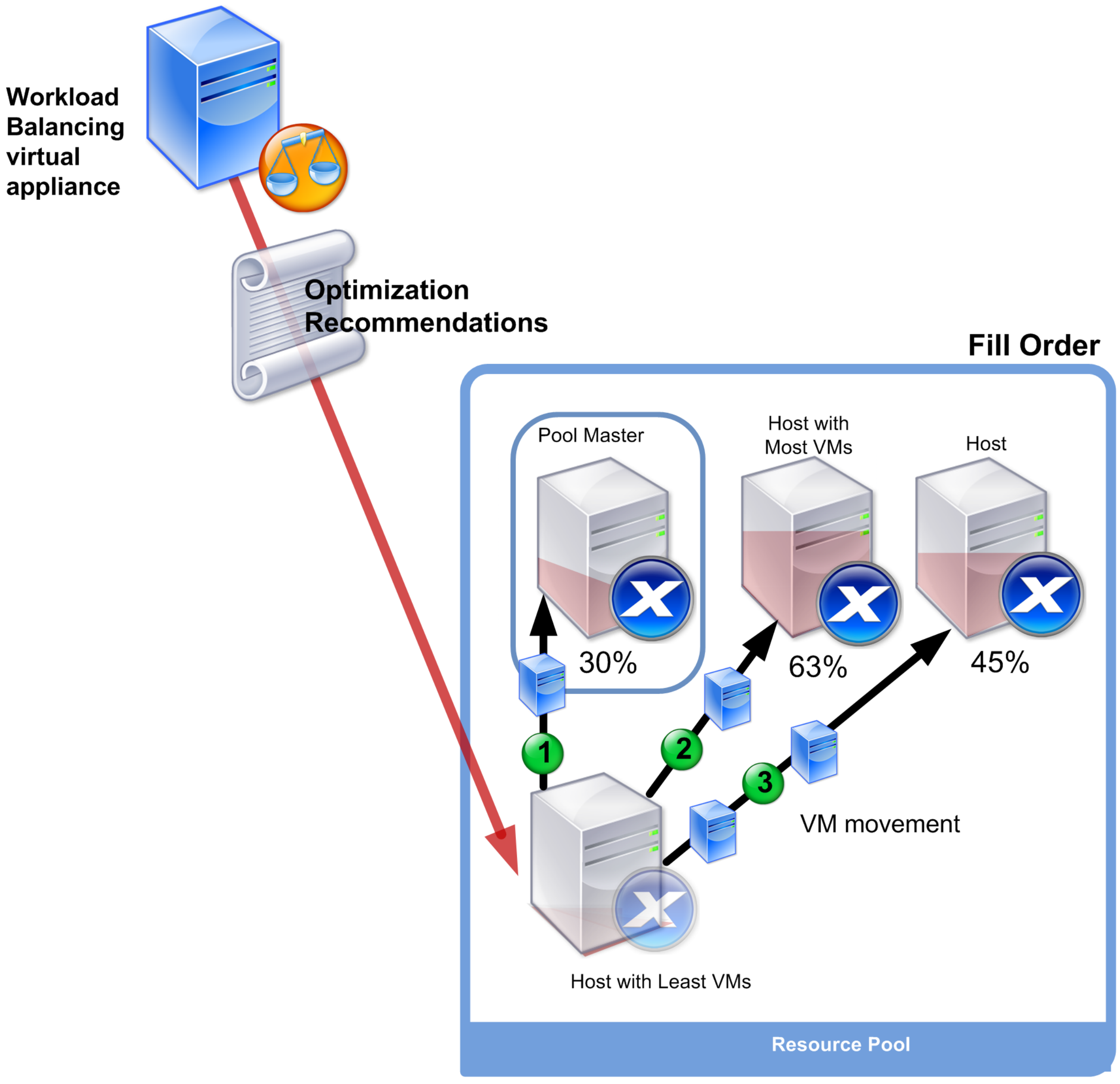
Understand power management behavior

Before Workload Balancing recommends powering hosts on or off, it selects the hosts to transfer VMs to. It does so in the following order:

1. Filling the pool coordinator since it is the host that cannot be powered off.
2. Filling the host with the most VMs.
3. Filling subsequent hosts according to which hosts have the most VMs running.

When Workload Balancing fills the pool coordinator, it does so assuming artificially low thresholds for the coordinator. Workload Balancing uses these low thresholds as a buffer to prevent the pool coordinator from being overloaded.

Workload Balancing fills hosts in this order to encourage density.



When Workload Balancing detects a performance issue while the pool is in Maximum Density mode, it recommends migrating workloads among the powered-on hosts. If Workload Balancing cannot resolve the issue using this method, it attempts to power on a host. Workload Balancing determines which hosts to power on by applying the same criteria that it would if the optimization mode was set to Maximum Performance.

When Workload Balancing runs in Maximum Performance mode, Workload Balancing recommends powering on hosts until the resource utilization on all pool members falls below the High threshold.

While migrating VMs, if Workload Balancing determines that increasing capacity benefits the overall performance of the pool, it powers on hosts automatically or recommends doing so.

> **Important:**
>
> Workload Balancing only recommends powering on a host that Workload Balancing powered off.

Design environments for power management and VM consolidation

When you are planning a XenServer implementation and you intend to configure automatic VM consolidation and power management, consider your workload design. For example, you might want to:

- Place different types of workloads in separate pools.

  If you have an environment with distinct types of workloads, consider whether to locate the VMs hosting these workloads in different pools. Also consider splitting VMs that host types of applications that perform better with certain types of hardware into different pool.

  Because power management and VM consolidation are managed at the pool level, design pools so they contain workloads that you want consolidated at the same rate. Ensure that you factor in considerations such as those discussed in Configure advanced settings.

- Exclude hosts from Workload Balancing.

  Some hosts might need to be always on. For more information, see Exclude hosts from recommendations.

Understand when Workload Balancing makes recommendations

Workload Balancing continuously evaluates the resource metrics of physical hosts and VMs across the pools that it is managing against thresholds. Thresholds are preset values that function like boundaries that a host must exceed before Workload Balancing can make an optimization recommendation. The Workload Balancing process is as follows:

1. Workload Balancing detects that the threshold for a resource was violated.
2. Workload Balancing evaluates if it makes an optimization recommendation.
3. Workload Balancing determines which hosts it recommends function as the destination hosts and in what order to make any optimizations. A destination host is the host where Workload Balancing recommends relocating one or more VMs.
4. Workload Balancing makes an optimization recommendation.

When evaluating hosts in the pool to make an optimization recommendation, Workload Balancing uses thresholds and weightings as follows:

- **Thresholds** are the boundary values that Workload Balancing compares the resource metrics of your pool against. The thresholds are used to determine whether to make a recommendation and what hosts are a suitable candidate for hosting relocated VMs.
- **Weightings** are a way of ranking resources according to how much you want them to be considered, are used to determine the processing order. After Workload Balancing decides to make a recommendation, it uses your specifications of which resources are important to determine the following:
  - Which hosts' performance to address first
  - Which VMs to recommend migrating first

For each resource Workload Balancing monitors, it has four levels of thresholds: Critical, High, Medium, and Low. Workload Balancing evaluates whether to make a recommendation when a resource metric on a host:

- Exceeds the High threshold when the pool is running in Maximum Performance mode (improve performance)
- Drops below the Low threshold when the pool is running in Maximum Density mode (consolidate VMs on hosts)
- Exceeds the Critical threshold when the pool is running in Maximum Density mode (improve performance)

If the High threshold for a pool running in Maximum Performance mode is 80%, when CPU utilization on a host reaches 80.1%, Workload Balancing evaluates whether to issue a recommendation.

When a resource violates its threshold, Workload Balancing evaluates the resource metric against historical performance to prevent making an optimization recommendation based on a temporary spike. To do so, Workload Balancing creates a historically averaged utilization metric by evaluating the data for resource utilization captured at the following times:

Data captured	Weight
Immediately, at the time threshold was exceeded. That is, real-time data.	70%
30 minutes before the threshold was exceeded	25%
24 hours before the threshold was exceeded	5%

If CPU utilization on the host exceeds the threshold at 12:02 PM, Workload Balancing checks the utilization at 11:32 AM that day, and at 12:02PM on the previous day. For example, if CPU utilization is at the following values, Workload Balancing doesn't make a recommendation:

- 80.1% at 12:02 PM that day

- 50% at 11:32 AM that day
- 78% at 12:32 PM the previous day

This behavior is because the historically averaged utilization is 72.5%, so Workload Balancing assumes that the utilization is a temporary spike. However, if the CPU utilization was 83% at 11:32AM, Workload Balancing makes a recommendation since the historically averaged utilization is 80.1%.

Optimization and consolidation process

The Workload Balancing process for determining potential optimizations varies according to the optimization mode - Maximum Performance or Maximum Density. However, regardless of the optimization mode, the optimization and placement recommendations are made using a two-stage process:

1. Determine potential optimizations: which VMs to migrate off hosts.
2. Determine placement recommendations: which hosts would be suitable candidates for new VMs.

> **Note:**
>
> Workload Balancing only recommends migrating VMs that meet the XenServer criteria for live migration. One of these criteria is that the destination host must have the storage the VM requires. The destination host must also have sufficient resources to accommodate adding the VM without exceeding the thresholds of the optimization mode configured on the pool. For example, the High threshold in Maximum Performance mode and the Critical threshold for Maximum Density mode.

When Workload Balancing is running in automated mode, you can tune the way it applies recommendations. For more information, see Set conservative or aggressive automated recommendations.

Optimization recommendation process in Maximum Performance mode When running in Maximum Performance mode, Workload Balancing uses the following process to determine potential optimizations:

1. Every two minutes Workload Balancing evaluates the resource utilization for each host in the pool. It does so by monitoring on each host and determining if each resource's utilization exceeds its High threshold. For more information, see Change the critical threshold.

   In Maximum Performance mode, if a utilization of a resource exceeds its High threshold, Workload Balancing starts the process to determine whether to make an optimization recommendation. Workload Balancing determines whether to make an optimization recommendation based on whether doing so can ease performance constraints, such as ones revealed by the High threshold.

For example, consider the case where Workload Balancing sees that insufficient CPU resources negatively affect the performance of the VMs on a host. If Workload Balancing can find another host with less CPU utilization, it recommends moving one or more VMs to another host.

2. If a resource's utilization on a host exceeds the relevant threshold, Workload Balancing combines the following data to form the historically averaged utilization:

   - The resource's current utilization
   - Historical data from 30 minutes ago
   - Historical data from 24 hours ago
     If the historically averaged utilization exceeds the threshold of a resource, Workload Balancing determines it makes an optimization recommendation.

3. Workload Balancing uses metric weightings to determine what hosts to optimize first. The resource to which you have assigned the most weight is the one that Workload Balancing attempts to address first. For more information, see Tune metric weightings.

4. Workload Balancing determines which hosts can support the VMs it wants to migrate off hosts.

   Workload Balancing makes this determination by calculating the projected effect on resource utilization of placing different combinations of VMs on hosts. Workload Balancing uses a method of performing these calculations that in mathematics is known as permutation.

   To do so, Workload Balancing creates a single metric or score to forecast the impact of migrating a VM to the host. The score indicates the suitability of a host as a home for more VMs.

   To score host performance, Workload Balancing combines the following metrics:

   - The current metrics of the host
   - The metrics of the host from the last 30 minutes
   - The metrics of the host from 24 hours ago
   - The metrics of the VM.

5. After scoring hosts and VMs, Workload Balancing attempts to build virtual models of what the hosts look like with different combinations of VMs. Workload Balancing uses these models to determine the best host to place the VM.

   In Maximum Performance mode, Workload Balancing uses metric weightings to determine what hosts to optimize first and what VMs on those hosts to migrate first. Workload Balancing bases its models on the metric weightings. For example, if CPU utilization is assigned the highest importance, Workload Balancing sorts hosts and VMs to optimize according to the following criteria:

   a) What hosts are running closest to the High threshold for CPU utilization.
   b) What VMs have the highest CPU utilization or are running the closest to its High threshold.

6. Workload Balancing continues calculating optimizations. It views hosts as candidates for optimization and VMs as candidates for migration until predicted resource utilization on the host hosting the VM drops below the High threshold. Predicted resource utilization is the resource utilization that Workload Balancing forecasts a host has after Workload Balancing has added or removed a VM from the host.

Consolidation process in Maximum Density mode Workload Balancing determines whether to make a recommendation based on whether it can migrate a VM onto a host and still run that host below the Critical threshold.

1. When a resource's utilization drops below its Low threshold, Workload Balancing begins calculating potential consolidation scenarios.
2. When Workload Balancing discovers a way that it can consolidate VMs on a host, it evaluates whether the destination host is a suitable home for the VM.
3. Like in Maximum Performance mode, Workload Balancing scores the host to determine the suitability of a host as a home for new VMs.

   Before Workload Balancing recommends consolidating VMs on fewer hosts, it checks that resource utilization on those hosts after VMs are relocated to them is below Critical thresholds.

   > **Note:**
   >
   > Workload Balancing does not consider metric weightings when it makes a consolidation recommendation. It only considers metric weightings to ensure performance on hosts.

4. After scoring hosts and VMs, Workload Balancing attempts to build virtual models of what the hosts look like with different combinations of VMs. It uses these models to determine the best host to place the VM.
5. Workload Balancing calculates the effect of adding VMs to a host until it forecasts that adding another VM causes a host resource to exceed the Critical threshold.
6. Workload Balancing recommendations always suggest filling the pool coordinator first since it is the host that cannot be powered off. However, Workload Balancing applies a buffer to the pool coordinator so that it cannot be over-allocated.
7. Workload Balancing continues to recommend migrating VMs on to hosts until all remaining hosts exceed a Critical threshold when a VM is migrated to them.

Change the critical thresholds

You might want to change critical thresholds as a way of controlling when optimization recommendations are triggered. This section provides guidance about:

- How to modify the default Critical thresholds on hosts in the pool
- How values set for Critical threshold alter High, Medium, and Low thresholds.

Workload Balancing determines whether to produce recommendations based on whether the averaged historical utilization for a resource on a host violates its threshold. Workload Balancing recommendations are triggered when the High threshold in Maximum Performance mode or Low and Critical thresholds for Maximum Density mode are violated. For more information, see Optimization and consolidation process.

After you specify a new Critical threshold for a resource, Workload Balancing resets the other thresholds of the resource relative to the new Critical threshold. To simplify the user interface, the Critical threshold is the only threshold you can change through XenCenter.

The following table shows the default values for the Workload Balancing thresholds:

Metric	Critical	High	Medium	Low
CPU Utilization	90%	76.5%	45%	22.5%
Free Memory	51 MB	63.75 MB	510 MB	1020 MB
Network Reads	25 MB/sec	21.25 MB/sec	12.5 MB/sec	6.25 MB/sec
Network Writes	25 MB/sec	21.25 MB/sec	12.5 MB/sec	6.25 MB/sec
Disk Reads	25 MB/sec	21.25 MB/sec	12.5 MB/sec	6.25 MB/sec
Disk Writes	25 MB/sec	21.25 MB/sec	12.5 MB/sec	6.25 MB/sec

To calculate the threshold values for all metrics except memory, Workload Balancing multiplies the new value for the Critical threshold with the following factors:

- **High Threshold Factor**: 0.85
- **Medium Threshold Factor**: 0.50
- **Low Threshold Factor**: 0.25

For example, if you increase the Critical threshold for CPU utilization to 95%, Workload Balancing resets the other thresholds as follows:

- High: 80.75%
- Medium: 47.5%
- Low: 23.75%

To calculate the threshold values for free memory, Workload Balancing multiplies the new value for the Critical threshold with these factors:

- **High Threshold Factor**: 1.25

- **Medium Threshold Factor**: 10.0
- **Low Threshold Factor**: 20.0

For example, if you increase the Critical threshold for free memory to 45 MB, Workload Balancing resets the other thresholds as follows:

- High: 56.25 MB
- Medium: 450 MB
- Low: 900 MB

To perform this calculation for a specific threshold, multiply the factor for the threshold with the value you entered for the critical threshold for that resource:

```
High, Medium, or Low Threshold = Critical Threshold * High, Medium, or
    Low Threshold Factor
```

While the Critical threshold triggers many optimization recommendations, other thresholds can also trigger optimization recommendations, as follows:

- **High threshold**.
  - **Maximum Performance**. Exceeding the High threshold triggers optimization recommendations to relocate a VM to a host with lower resource utilization.
  - **Maximum Density**. Workload Balancing doesn't recommend placing a VM on host when moving that VM to the host causes the host resource utilization to exceed a High threshold.
- **Low threshold**.
  - **Maximum Performance**. Workload Balancing does not trigger recommendations from the Low threshold.
  - **Maximum Density**. When a metric value drops below the Low threshold, Workload Balancing determines that hosts are underutilized and makes an optimization recommendation to consolidate VMs on fewer hosts. Workload Balancing continues to recommend moving VMs onto a host until the metric values for one of the host's resources reaches its High threshold.

    However, after a VM is relocated, utilization of a resource on the VM's new host can exceed a Critical threshold. In this case, Workload Balancing temporarily uses an algorithm similar to the Maximum Performance load-balancing algorithm to find a new host for the VMs. Workload Balancing continues to use this algorithm to recommend moving VMs until resource utilization on hosts across the pool falls below the High threshold.

To change the critical thresholds:

1. In XenCenter, select your pool.

2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, select **Critical Thresholds**. These critical thresholds are used to evaluate host resource utilization.
5. In the **Critical Thresholds** page, type one or more new values in the **Critical Thresholds** boxes. The values represent resource utilization on the host.

   Workload Balancing uses these thresholds when making VM placement and pool-optimization recommendations. Workload Balancing strives to keep resource utilization on a host below the critical values set.

Tune metric weightings

How Workload Balancing uses metric weightings when determining which hosts and VMs to process first varies according to the optimization mode: Maximum Density or Maximum Performance. In general, metric weightings are used when a pool is in Maximum Performance mode. However, when Workload Balancing is in Maximum Density mode, it does use metric weightings when a resource exceeds its Critical threshold.

When Workload Balancing is processing optimization recommendations, it creates an optimization order. Workload Balancing determines the order by ranking the hosts according to which hosts have the highest metric values for whatever resource is ranked as the most important in the metric weightings page.

Maximum Performance mode

In Maximum Performance mode, Workload Balancing uses metric weightings to determine:

- On which hosts to first address the performance
- Which VMs to recommend migrating first

For example, if Network Writes is the most important resource, Workload Balancing first makes optimization recommendations for the host with the highest number of Network Writes per second. To make Network Writes the most important resource move the **Metric Weighting** slider to the right and all the other sliders to the middle.

If you configure all resources to be equally important, Workload Balancing addresses CPU utilization first and memory second, as these resources are typically the most constrained. To make all resources equally important, set the **Metric Weighting** slider is in the same place for all resources.

Maximum Density mode

In Maximum Density mode, Workload Balancing only uses metric weightings when a host reaches the Critical threshold. At that point, Workload Balancing applies an algorithm similar to the algorithm for Maximum Performance until no hosts exceed the Critical thresholds. When using this algorithm, Workload Balancing uses metric weightings to determine the optimization order in the same way as it does for Maximum Performance mode.

If two or more hosts have resources exceeding their Critical thresholds, Workload Balancing verifies the importance you set for each resource. It uses this importance to determine which host to optimize first and which VMs on that host to relocate first.

For example, your pool contains host A and host B, which are in the following state:

- The CPU utilization on host A exceeds its Critical threshold and the metric weighting for CPU utilization is set to **More Important**.
- The memory utilization on host B exceeds its Critical threshold and the metric weighting for memory utilization is set to **Less Important**.

Workload Balancing recommends optimizing host A first because the resource on it that reached the Critical threshold is the resource assigned the highest weight. After Workload Balancing determines that it must address the performance on host A, Workload Balancing then begins recommending placements for VMs on that host. It begins with the VM that has the highest CPU utilization, since that CPU utilization is the resource with the highest weight.

After Workload Balancing has recommended optimizing host A, it makes optimization recommendations for host B. When it recommends placements for the VMs on host B, it does so by addressing CPU utilization first, since CPU utilization was assigned the highest weight. If there are more hosts that need optimization, Workload Balancing addresses the performance on those hosts according to what host has the third highest CPU utilization.

By default, all metric weightings are set to the farthest point on the slider: More Important.

> **Note:**
>
> The weighting of metrics is relative. If all metrics are set to the same level, even if that level is Less Important, they are all be weighted the same. The relation of the metrics to each other is more important than the actual weight at which you set each metric.

To edit metric weighting factors

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.

4. In the left pane, select **Metric Weighting**.
5. In **Metric Weighting** page, as desired, adjust the sliders beside the individual resources.

   Move the slider towards **Less Important** to indicate that ensuring VMs always have the highest available amount of this resource is not as vital for this pool.

Exclude hosts from recommendations

When configuring Workload Balancing, you can specify that specific physical hosts are excluded from Workload Balancing optimization and placement recommendations, including Start On placement recommendations.

Situations when you might want to exclude hosts from recommendations include when:

- You want to run the pool in Maximum Density mode and consolidate and shut down hosts, but you want to exclude specific hosts from this behavior.
- You have two VM workloads that must always run on the same host. For example, if the VMs have complementary applications or workloads.
- You have workloads that you do not want moved: for example, a domain controller or database server.
- You want to perform maintenance on a host and you do not want VMs placed on the host.
- The performance of the workload is so critical that the cost of dedicated hardware is irrelevant.
- Specific hosts are running high-priority workloads, and you do not want to use the HA feature to prioritize these VMs.
- The hardware in the host is not the optimum for the other workloads in the pool.

Regardless of whether you specify a fixed or scheduled optimization mode, excluded hosts remain excluded even when the optimization mode changes. Therefore, if you only want to prevent Workload Balancing from shutting off a host automatically, consider disabling Power Management for that host instead. For more information, see Optimize and manage power automatically.

When you exclude a host from recommendations, you are specifying for Workload Balancing not to manage that host at all. This configuration means that Workload Balancing doesn't make any optimization recommendations for an excluded host. In contrast, when you don't select a host to participate in Power Management, Workload Balancing manages the host, but doesn't make power management recommendations for it.

To exclude hosts from Workload Balancing

Use this procedure to exclude a host in a pool that Workload Balancing is managing from power management, host evacuation, placement, and optimization recommendations.

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, select **Excluded Hosts**.
5. In **Excluded Hosts** page, select the hosts for which you do not want Workload Balancing to recommend alternate placements and optimizations.

Configure advanced settings

Workload Balancing supplies some advanced settings that let you control how Workload Balancing applies automated recommendations. These settings appear on the **Advanced** page of the Workload Balancing Configuration dialog. To get to the **Advanced** page, complete the following steps:

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, select **Advanced**.

The following sections describe the behaviors that can be configured in the **Advanced** settings.

Set conservative or aggressive automated recommendations

When running in automated mode, the frequency of optimization and consolidation recommendations and how soon they are automatically applied is a product of multiple factors, including:

- How long you specify Workload Balancing waits after moving a VM before making another recommendation
- The number of recommendations Workload Balancing must make before applying a recommendation automatically
- The severity level a recommendation must achieve before the optimization is applied automatically
- The level of consistency in recommendations (recommended VMs to move, destination hosts) Workload Balancing requires before applying recommendations automatically

In general, only adjust the settings for these factors in the following cases:

- You have guidance from XenServer Technical Support
- You made significant observation and testing of the behavior of your pool with Workload Balancing enabled

Incorrectly configuring these settings can result in Workload Balancing not making any recommendations.

VM migration interval

You can specify the number of minutes Workload Balancing waits after the last time a VM was moved, before Workload Balancing can make another recommendation for that VM. The recommendation interval is designed to prevent Workload Balancing from generating recommendations for artificial reasons, for example, if there was a temporary utilization spike.

When automation is configured, it is especially important to be careful when modifying the recommendation interval. If an issue occurs that leads to continuous, recurring spikes, decreasing the interval can generate many recommendations and, therefore, relocations.

> **Note:**
>
> Setting a recommendation interval does not affect how long Workload Balancing waits to factor recently rebalanced hosts into recommendations for Start-On Placement, Resume, and Maintenance Mode.

Recommendation count

Every two minutes, Workload Balancing checks to see if it can generate recommendations for the pool it is monitoring. When you enable automation, you can specify the number of times a consistent recommendation must be made before Workload Balancing automatically applies the recommendation. To do so, you configure a setting known as the **Recommendation Count**, as specified in the **Recommendations** field. The **Recommendation Count** and the **Optimization Aggressiveness** setting let you fine-tune the automated application of recommendations in your environment.

Workload Balancing uses the similarity of recommendations to make the following checks:

1. Whether the recommendation is truly needed
2. Whether the destination host has stable enough performance over a prolonged period to accept a relocated VM without needing to move it off the host again shortly

Workload Balancing uses the Recommendation Count value to determine whether a recommendation must be repeated before Workload Balancing automatically applies the recommendation. Workload Balancing uses this setting as follows:

1. Every time Workload Balancing generates a recommendation that meets its consistency requirements, as indicated by the Optimization Aggressiveness setting, Workload Balancing increments the Recommendation Count. If the recommendation does not meet the consistency requirements, Workload Balancing might reset the Recommendation Count to zero. This behavior depends on the factors described in Optimization aggressiveness.

2. When Workload Balancing generates enough consistent recommendations to meet the value for the Recommendation Count, as specified in the **Recommendations** field, it automatically applies the recommendation.

If you choose to modify this setting, the value to set varies according to your environment. Consider these scenarios:

- If host loads and activity increase rapidly in your environment, you might want to increase value for the Recommendation Count. Workload Balancing generates recommendations every two minutes. For example, if you set this interval to **3**, then six minutes later Workload Balancing applies the recommendation automatically.
- If host loads and activity increase gradually in your environment, you might want to decrease the value for the Recommendation Count.

Accepting recommendations uses system resources and affects performance when Workload Balancing is relocating the VMs. Increasing the Recommendation Count increases the number of matching recommendations that must occur before Workload Balancing applies the recommendation. This setting encourages Workload Balancing to apply more conservative, stable recommendations and can decrease the potential for spurious VM moves. The Recommendation Count is set to a conservative value by default.

Because of the potential impact adjusting this setting can have on your environment, only change it with extreme caution. Preferably, make these adjustments by testing and iteratively changing the value or under the guidance of XenServer Technical Support.

Recommendation severity

All optimization recommendations include a severity rating (Critical, High, Medium, Low) that indicates the importance of the recommendation. Workload Balancing bases this rating on a combination of factors including the following:

- Configuration options you set, such as thresholds and metric tunings
- Resources available for the workload
- Resource-usage history.

The severity rating for a recommendation appears in the **Optimization Recommendations** pane on the **WLB** tab.

When you configure Workload Balancing to apply recommendations automatically, you can set the minimum severity level to associate with a recommendation before Workload Balancing automatically applies it.

Optimization aggressiveness

To provide extra assurance when running in automated mode, Workload Balancing has consistency criteria for accepting optimizations automatically. This criteria can help to prevent moving VMs due to spikes and anomalies. In automated mode, Workload Balancing does not accept the first recommendation it produces. Instead, Workload Balancing waits to apply a recommendation automatically until a host or VM exhibits consistent behavior over time. Consistent behavior over time includes factors like whether a host continues to trigger recommendations and whether the same VMs on that host continue to trigger recommendations.

Workload Balancing determines if behavior is consistent by using criteria for consistency and by having criteria for the number of times the same recommendation is made. You can configure how strictly you want Workload Balancing to apply the consistency criteria using the **Optimization Aggressiveness** setting. You can use this setting to control the amount of stability you want in your environment before Workload Balancing applies an optimization recommendation. The most stable setting, Low aggressiveness, is configured by default. In this context, the term stable means the similarity of the recommended changes over time, as explained throughout this section. Aggressiveness is not desirable in most environments. Therefore, Low is the default setting.

Workload Balancing uses up to four criteria to ascertain consistency. The number of criteria that must be met varies according to the level you set in the **Optimization Aggressiveness** setting. The lower the level (for example, Low or Medium) the less aggressive Workload Balancing is in accepting a recommendation. In other words, Workload Balancing is stricter about requiring criteria to match when aggressiveness is set to Low.

For example, if the aggressiveness level is set to Low, each criterion for Low must be met the number of times specified by the Recommendation Count value before automatically applying the recommendation.

If you set the Recommendation Count to **3**, Workload Balancing waits until all the criteria listed for Low are met and repeated in three consecutive recommendations. This setting helps ensure that the VM actually needs to be moved and that the recommended destination host has stable resource utilization over a longer period. It reduces the potential for a recently moved VM to be moved off a host due to host performance changes after the move. By default, this setting is set to Low to encourage stability.

We do not recommend increasing the **Optimization Aggressiveness** setting to increase the frequency with which your hosts are being optimized. If you think that your hosts aren't being optimized quickly or frequently enough, try adjusting the Critical thresholds. Compare the thresholds against the Pool Health report.

The consistency criteria associated with the different levels of aggressiveness is the following:

Low:

- All VMs in subsequent recommendations must be the same (as demonstrated by matching UUIDs in each recommendation).
- All destination hosts must be the same in subsequent recommendations
- The recommendation that immediately follows the initial recommendation must match or else the Recommendation Count reverts to 1

Medium:

- All VMs in subsequent recommendations must be from the same host; however, they can be different VMs from the ones in the first recommendation.
- All destination hosts must be the same in subsequent recommendations
- One of the next two recommendations that immediately follows the first recommendation must match or else the Recommendation Count reverts to 1

High:

- All VMs in the recommendations must be from the same host. However, the recommendations do not have to follow each other immediately.
- The host from which Workload Balancing recommended that the VM move must be the same in each recommendation
- The Recommendation Count remains at the same value even when the two recommendations that follow the first recommendation do not match

Optimization Aggressiveness example The following example illustrates how Workload Balancing uses the **Optimization Aggressiveness** setting and the Recommendation Count to determine whether to accept a recommendation automatically.

Each optimization recommendation issued by Workload Balancing proposes three VM placements. After these proposed placements, the recommendation count associated with each aggressiveness level is the number of times there have been consecutive recommendation at that Optimization Aggressiveness setting.

In the following examples, when the **Optimization Aggressiveness** setting is set to High, the Recommendation Count continues to increase after Recommendation 1, 2, and 3. This increase happens even though the same VMs are not recommended for new placements in each recommendation. Workload Balancing applies the placement recommendation with Recommendation 3 because it has seen the same behavior from that host for three consecutive recommendations.

In contrast, when set to Low aggressiveness, the consecutive recommendations count does not increase for the first four recommendations. The Recommendation Count resets to 1 with each recommendation because the same VMs were not recommended for placements. The Recommendation Count does not start to increase until the same recommendation is made in Recommendation #5. Finally, Workload Balancing automatically applies the recommendation made in Recommendation #6 after the third time it issues the same placement recommendations.

Recommendation 1:

Proposed placements:

- Move VM1 from host A to host B
- Move VM3 from host A to host B
- Move VM5 from host A to host C

Recommendation counts:

- High Aggressiveness Recommendation Count: 1
- Medium Aggressiveness Recommendation Count: 1
- Low Aggressiveness Recommendation Count: 1

Recommendation 2:

Proposed placements:

- Move VM1 from host A to host B
- Move VM3 from host A to host C
- Move VM7 from host A to host C

Recommendation counts:

- High Aggressiveness Recommendation Count: 2
- Medium Aggressiveness Recommendation Count: 1
- Low Aggressiveness Recommendation Count: 1

Recommendation 3:

Proposed placements:

- Move VM1 from host A to host B
- Move VM3 from host A to host C
- Move VM5 from host A to host C

Recommendation counts:

- High Aggressiveness Recommendation Count: 3 (Apply)
- Medium Aggressiveness Recommendation Count: 1
- Low Aggressiveness Recommendation Count: 1

Recommendation 4:

Proposed placements:

- Move VM1 from host A to host B
- Move VM3 from host A to host B
- Move VM5 from host A to host C

Recommendation counts:

- Medium Aggressiveness Recommendation Count: 2
- Low Aggressiveness Recommendation Count: 1

Recommendation 5:

Proposed placements:

- Move VM1 from host A to host B
- Move VM3 from host A to host B
- Move VM5 from host A to host C

Recommendation counts:

- Medium Aggressiveness Recommendation Count: 3 (Apply)
- Low Aggressiveness Recommendation Count: 2

Recommendation 6:

Proposed placements:

- Move VM1 from host A to host B
- Move VM3 from host A to host B
- Move VM5 from host A to host C

Recommendation counts:

- Low Aggressiveness Recommendation Count: 3 (Apply)

To configure VM recommendation intervals

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, click **Advanced**.
5. In the **VM Recommendation Interval** section, do one or more of the following:
    - In the **Minutes** box, type a value for the number of minutes Workload Balancing waits before making another optimization recommendation on a newly rebalanced host.

- In the **Recommendations** box, type a value for the number of recommendations you want Workload Balancing to make before it applies a recommendation automatically.
- Select a minimum severity level before optimizations are applied automatically.
- Modify how aggressively Workload Balancing applies optimization recommendations when it is running in automated mode. Increasing the aggressiveness level reduces constraints on the consistency of recommendations before automatically applying them. The **Optimization Aggressiveness** setting directly complements the **Recommendations** setting: that is, the recommendations count.

  > **Note:**
  >
  > If you type "1"for the value in the **Recommendations** setting, the **Optimization Aggressiveness** setting is not relevant.

Adjust the Pool Audit Trail granularity settings

Follow this procedure to modify the granularity settings:

1. In XenCenter, select your pool.
2. In the **Properties** pane of the pool, click the **WLB** tab.
3. In the **WLB** tab, click **Settings**.
4. In the left pane, click **Advanced**.
5. On the **Advanced** page, click the **Pool Audit Trail Report Granularity** list, and select an option from the list.

   > **Important:**
   >
   > Select the granularity based on your audit log requirements. For example, if you set your audit log report granularity to Minimum, the report only captures limited amount of data for specific users and object types. If you set the granularity to Medium, the report provides a user-friendly report of the audit log. If you choose to set the granularity to Maximum, the report contains detailed information about the audit log report. Setting the audit log report to Maximum can cause the Workload Balancing server to use more disk space and memory.

6. To confirm your changes, click **OK**.

View Pool Audit Trail reports based on objects in XenCenter

Follow this procedure to run and view reports of Pool Audit Trail based on the selected object:

1. After you have set the Pool Audit Trail Granularity setting, click **Reports**. The Workload Reports page appears.
2. Select **Pool Audit Trail** on the left pane.
3. You can run and view the reports based on a specific Object by choosing it from the **Object** list. For example, choose **Host** from the list to get the reports based on host alone.

Customize the event objects and actions captured by the Pool Audit Trail

To customize the event objects and actions captured by the Pool Audit Trail, you must sign in to the PostgreSQL database on the Workload Balancing virtual appliance, make the relevant changes to the list of event objects or actions, and then restart the Workload Balancing virtual appliance.

Sign in to the PostgreSQL database

1. Log on to the Workload Balancing virtual appliance console.
2. Run the following command:

   ```
   psql -Upostgres -dWorkloadBalancing
   <!--NeedCopy-->
   ```

3. Enter the database password. You set the database password when you ran the Workload Balancing configuration wizard after you imported the virtual appliance.

Customize event objects

> **Note:**
>
> In the command syntax that follows, `event_object` represents the name of the event object you want to add, update, or disable.

Enable an event object:

```
select * from update_audit_log_objects('event_object', true);
<!--NeedCopy-->
```

Disable an event object:

```
select * from update_audit_log_objects('event_object', false);
<!--NeedCopy-->
```

Get a list of event objects that are currently disabled:

```
select * from hv_audit_log_get_event_objects(false);
<!--NeedCopy-->
```

Get a list of event objects that are currently enabled:

```
select * from hv_audit_log_get_event_objects(true);
<!--NeedCopy-->
```

Customize event actions

> **Note:**
>
> In the command syntax that follows, `event_action` represents the name of the event action you want to add, update, or disable.

Enable an event action:

```
select * from update_audit_log_actions('event_action', true);
<!--NeedCopy-->
```

Disable an event action:

```
select * from update_audit_log_actions('event_action', false);
<!--NeedCopy-->
```

Get a list of event actions that are currently disabled:

```
select * from hv_audit_log_get_event_actions(false);
<!--NeedCopy-->
```

Get a list of event actions that are currently enabled:

```
select * from hv_audit_log_get_event_actions(true);
<!--NeedCopy-->
```

Restart the Workload Balancing virtual appliance Run the following commands to quit PostgreSQL and restart the Workload Balancing virtual appliance.

```
\q
<!--NeedCopy-->
```

```
systemctl restart workloadbalancing
<!--NeedCopy-->
```

Set alert level for Workload Balancing alerts in XenCenter

You can set the alert level for Workload Balancing alerts in XenCenter by using the Management API.

Complete the following steps:

1. Run the following command on the pool coordinator to set the alert level for each alert code:

```
xe pool-send-wlb-configuration config:<wlb-alert-code>=<alert-
    level>
<!--NeedCopy-->
```

The 4 `wlb-alert-code` types are:

- MESSAGE_PRIORITY_WLB_OPTIMIZATION_ALERT - If Workload Balancing gives an optimization recommendation, this alert is raised.
- MESSAGE_PRIORITY_WLB_VM_RELOCATION - If Workload Balancing relocates a VM to other host, this alert is raised.
- MESSAGE_PRIORITY_WLB_HOST_POWER_OFF - If Workload Balancing optimization mode has been configured to `Maximize Density` and a host is powered off because there are no VMs running on the host, this alert is raised.
- MESSAGE_PRIORITY_WLB_HOST_POWER_ON - If Workload Balancing optimization mode has been configured to `Maximize Performance` and a host is powered on because doing so improves the host performance, this alert is raised.

The 6 `alert-level` types are:

- 0 - Mute the alert
- 1 - Critical
- 2 - Major
- 3 - Warning
- 4 - Minor
- 5 - Informational

2. Run the following command on the pool coordinator to view the alert levels set for the alert codes:

```
xe pool-retrieve-wlb-configuration
<!--NeedCopy-->
```

3. To test the alerts, raise a Workload Balancing alert and then click the `Notifications` panel to view the alert.

Administer Workload Balancing

May 26, 2023

After Workload Balancing has been running for a while, there are routine tasks that you might need to perform to keep Workload Balancing running optimally. You might need to perform these tasks because of changes to your environment (such as different IP addresses or credentials), hardware upgrades, or routine maintenance.

Connect to the Workload Balancing virtual appliance

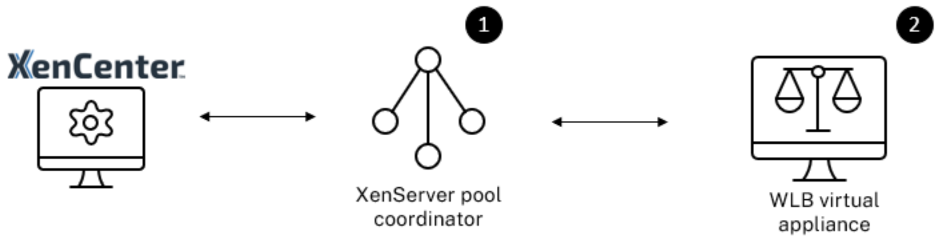
After Workload Balancing configuration, connect the pool you want managed to the Workload Balancing virtual appliance using either the CLI or XenCenter. Likewise, you might need to reconnect to the same virtual appliance at some point.

To connect a pool to your Workload Balancing virtual appliance, you need the following information:

- IP address or FQDN of the Workload Balancing virtual appliance
  - To obtain the IP address for the Workload Balancing virtual appliance:
    1. In XenCenter, go to the Workload Balancing virtual appliance **Console** tab.
    2. Log in as `root` with the root password you created when you imported the appliance.
    3. Run the following command: `ifconfig`.
  - To specify the Workload Balancing FQDN when connecting to the Workload Balancing server, first add its host name and IP address to your DNS server.
- The port number of the Workload Balancing virtual appliance. By default, XenServer connects to Workload Balancing on port 8012.

  Only edit the port number when you have changed it during Workload Balancing Configuration. The port number specified during Workload Balancing Configuration, in any firewall rules, and in the Connect to WLB Server dialog must match.
- Credentials for the resource pool you want Workload Balancing to monitor.
- Credentials for the Workload Balancing account you created during Workload Balancing configuration.

  This account is often known as the Workload Balancing user account. XenServer uses this account to communicate with Workload Balancing. You created this account on the Workload Balancing virtual appliance during Workload Balancing Configuration.



When you first connect to Workload Balancing, it uses the default thresholds and settings for balancing workloads. Automatic features, such as automated optimization mode, power management, and automation, are disabled by default.

Working with certificates

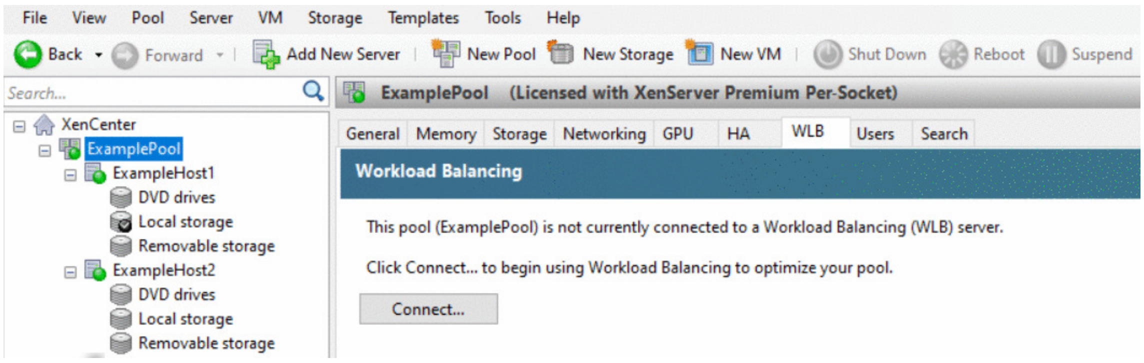
If you want to upload a different (trusted) certificate or configure certificate verification, note the following before connecting your pool to Workload Balancing:

- If you want XenServer to verify the self-signed Workload Balancing certificate, you must use the Workload Balancing IP address to connect to Workload Balancing. The self-signed certificate is issued to Workload Balancing based on its IP address.
- If you want to use a certificate from a certificate authority, it is easier to specify the FQDN when connecting to Workload Balancing. However, you can specify a static IP address in the **Connect to WLB Server** dialog. Use this IP address as the Subject Alternative Name (SAN) in the certificate.

For more information, see Certificates.

To connect your pool to the Workload Balancing virtual appliance

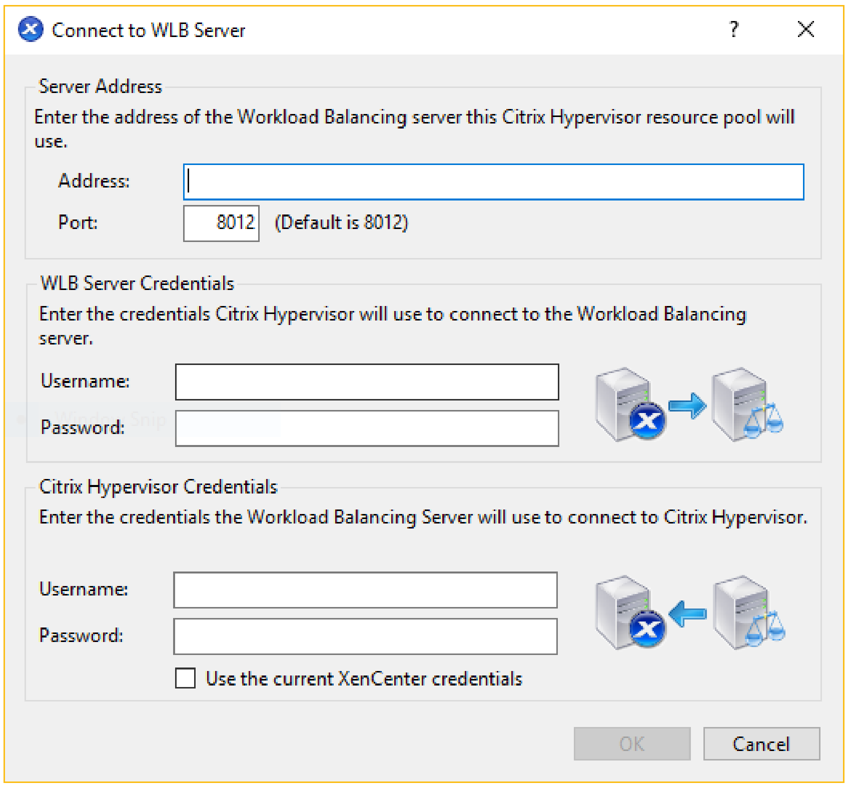
1. In XenCenter, select your resource pool and in its **Properties** pane, click the **WLB** tab. The **WLB** tab displays the **Connect** button.


File View Pool Server VM Storage Templates Tools Help
Back Forward Add New Server New Pool New Storage New VM Shut Down Reboot Suspend
Search...
XenCenter
ExamplePool
ExampleHost1
DVD drives
Local storage
Removable storage
ExampleHost2
DVD drives
Local storage
Removable storage
ExamplePool (Licensed with XenServer Premium Per-Socket)
General Memory Storage Networking GPU HA WLB Users Search
Workload Balancing
Connect...

This pool (ExamplePool) is not currently connected to a Workload Balancing (WLB) server.

Click Connect... to begin using Workload Balancing to optimize your pool.

2. In the **WLB** tab, click **Connect**. The **Connect to WLB Server** dialog box appears.



Connect to WLB Server

Server Address

Enter the address of the Workload Balancing server this Citrix Hypervisor resource pool will use.

Address:

Port: 8012 (Default is 8012)

WLB Server Credentials

Enter the credentials Citrix Hypervisor will use to connect to the Workload Balancing server.

Username:

Password:

Citrix Hypervisor Credentials

Enter the credentials the Workload Balancing Server will use to connect to Citrix Hypervisor.

Username:

Password:

☐ Use the current XenCenter credentials

OK Cancel

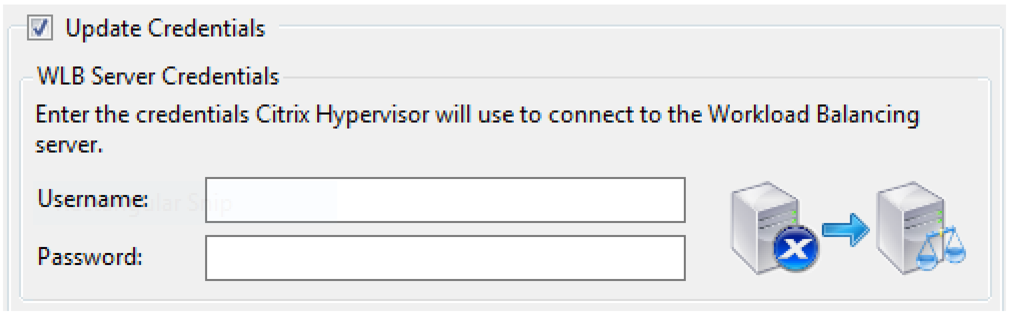
3. In the **Server Address** section, enter the following:

    a) In the **Address** box, type the IP address or FQDN of the Workload Balancing virtual appliance. For example, `WLB-appliance-computername.yourdomain.net`.

    b) (Optional) If you changed the Workload Balancing port during Workload Balancing Configuration, enter the port number in the **Port** box. XenServer uses this port to communicate with Workload Balancing.

       By default, XenServer connects to Workload Balancing on port 8012.

4. In the **WLB Server Credentials** section, enter the user name and password that the pool uses to connect to the Workload Balancing virtual appliance.



☑ Update Credentials

WLB Server Credentials

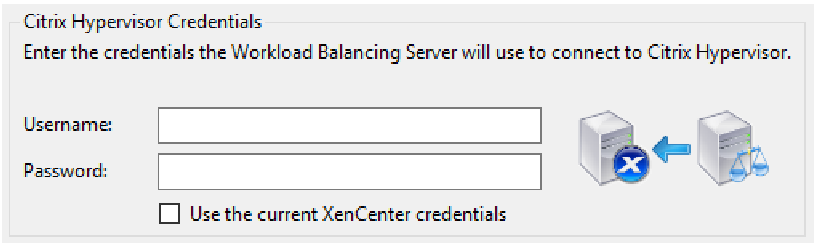
Enter the credentials Citrix Hypervisor will use to connect to the Workload Balancing server.

Username:

Password:

These credentials must be for the account you created during Workload Balancing configuration. By default, the user name for this account is `wlbuser`.

5. In the **Citrix Hypervisor Credentials** section, enter the user name and password for the pool you are configuring. Workload Balancing uses these credentials to connect to the hosts in that pool.



Citrix Hypervisor Credentials
Enter the credentials the Workload Balancing Server will use to connect to Citrix Hypervisor.

Username:

Password:

☐ Use the current XenCenter credentials

To use the credentials with which you are currently logged into XenServer, select **Use the current XenCenter credentials**. If you have assigned a role to this account using the role-based access control (RBAC) feature, ensure that the role has sufficient permissions to configure Workload Balancing. For more information, see Workload Balancing Access Control Permissions.

After connecting the pool to the Workload Balancing virtual appliance, Workload Balancing automatically begins monitoring the pool with the default optimization settings. If you want to modify these settings or change the priority given to resources, wait until the XenCenter Log shows that discovery is finished before proceeding.

> **Important:**
>
> After Workload Balancing is running for a time, if you do not receive optimal recommendations, evaluate your performance thresholds as described in Configure Workload Balancing behavior. It is critical to set Workload Balancing to the correct thresholds for your environment or its recommendations might not be appropriate.

Workload Balancing access control permissions

When Role Based Access Control (RBAC) is implemented in your environment, all user roles can display the **WLB** tab. However, not all roles can perform all operations. The following table lists the minimum role administrators require to use Workload Balancing features:

Permission	Minimum Required Role
Configure, Initialize, Enable, Disable WLB	Pool Operator
Apply WLB Optimization Recommendations in WLB tab	Pool Operator
Modify WLB report subscriptions	Pool Operator
Accept WLB Placement Recommendations	VM Power Admin

Permission	Minimum Required Role
Generate WLB Reports, including the Pool Audit Trail report	Read Only
Display WLB Configuration	Read Only

The following table provides more details about permissions.

Permission	Allows Assignee To
Configure, Initialize, Enable, Disable WLB	Configure WLB Initialize WLB and change WLB servers Enable WLB Disable WLB
Apply WLB Optimization Recommendations in WLB tab	Apply any optimization recommendations that appear in the **WLB** tab
Modify WLB report subscriptions	Change the WLB report generated or its recipient
Accept WLB Placement Recommendations	Select one of the hosts Workload Balancing recommends for placement
Generate WLB Reports, including the Pool Audit Trail report	View and run WLB reports, including the Pool Audit Trail report
Display WLB Configuration	View WLB settings for a pool as shown on the WLB tab

If a user tries to use Workload Balancing and that user doesn't have sufficient permissions, a role elevation dialog appears. For more information about RBAC, see Role-based access control.

Reconfigure a pool to use another Workload Balancing virtual appliance

You can reconfigure a resource pool to use a different Workload Balancing virtual appliance.

If you are moving from an older version of the Workload Balancing virtual appliance to the latest version, before disconnecting your old virtual appliance, you can migrate its data to the new version of the virtual appliance. For more information, see Migrate data from an existing virtual appliance.

After disconnecting a pool from the old Workload Balancing virtual appliance, you can connect the pool by specifying the name of the new Workload Balancing virtual appliance.

To use a different Workload Balancing virtual appliance:

1. (Optional) Migrate data from an older version of the virtual appliance. For more information, see Migrate data from an existing virtual appliance.
2. In XenCenter, from the **Pool** menu, select **Disconnect Workload Balancing Server** and click **Disconnect** when prompted.
3. In the **WLB** tab, click **Connect**. The **Connect to WLB Server** dialog appears.
4. Connect to the new virtual appliance. For more information, see Connect to the Workload Balancing virtual appliance

Change the Workload Balancing credentials

After initial configuration, if you want to update the credentials XenServer and the Workload Balancing appliance use to communicate, use the following process:

1. To pause Workload Balancing, go to the **WLB** tab and click **Pause**.
2. Change the Workload Balancing credentials by running the `wlbconfig` command. For more information, see Workload Balancing Commands.
3. Re-enable Workload Balancing and specify the new credentials.
4. After the progress bar completes, click **Connect**.

   The **Connect to WLB Server** dialog box appears.
5. Click **Update Credentials**.
6. In the **Server Address** section, modify the following settings as appropriate:

   - In the **Address** box, type the IP address or FQDN of the Workload Balancing appliance.
   - (Optional.) If you changed the port number during Workload Balancing Configuration, enter that port number. The port number you specify in this box and during Workload Balancing Configuration is the port number XenServer uses to connect to Workload Balancing.

     By default, XenServer connects to Workload Balancing on port 8012.

     > **Note:**
     >
     > Only edit this port number if you changed it when you ran the Workload Balancing Configuration wizard. The port number value specified when you ran the Workload Balancing Configuration wizard and the Connect to WLB Server dialog must match.

7. In the **WLB Server Credentials** section, enter the user name (for example, `wlbuser`) and password the computers running XenServer uses to connect to the Workload Balancing server.

8. In the **Citrix Hypervisor Credentials** section, enter the user name and password for the pool you are configuring (typically the password for the pool coordinator). Workload Balancing uses these credentials to connect to the computers running XenServer in that pool.

9. In the **Citrix Hypervisor Credentials** section, enter the user name and password for the pool you are configuring. Workload Balancing uses these credentials to connect to the computers running XenServer in that pool.

   To use the credentials with which you are currently logged into XenServer, select **Use the current XenCenter credentials**.

Change the Workload Balancing IP address

To change the Workload Balancing IP address, complete the following:

1. To view the current Workload Balancing IP address, run the `ifconfig` command on the virtual appliance.
2. Open the `/etc/sysconfig/network-scripts/ifcfg-eth0` file by using an editing tool like vi.
3. To change the protocol from dhcp to static, change `BOOTPROTO=dhcp` to `BOOTPROTO=static`.
4. At the bottom of the file, set the IP address, netmask, gateway, and DNS addresses. For example:

   ```
   IPADDR=192.168.1.100
   NETMASK=255.255.255.0
   GATEWAY=192.168.1.1
   DNS1=1.1.1.1
   DNS2=8.8.8.8
   <!--NeedCopy-->
   ```

   > **Note:**
   >
   > Add as many DNS entries as you need.

5. Save and close the file.
6. For the changes to take effect, you must restart the networking system by running `systemctl restart network`.
7. Once the networking system has restarted, run the `ifconfig` command again to view the new Workload Balancing IP address.
8. To check that the Workload Balancing service is running normally, run the `systemctl status workloadbalancing` command.

If the returned result contains `Active: active (running)`, the Workload Balancing service is running normally. If the result contains `Active: inactive (dead)` or any other status, the Workload Balancing might exit abnormally.

Change the Workload Balancing virtual appliance configuration

When you first install the Workload Balancing virtual appliance it has the following default configuration:

Configuration	Value
Number of vCPUs	2
Memory (RAM)	2 GB
Disk space	30 GB

These values are suitable for most environments. If you are monitoring very large pools, you might consider increasing these values.

Change the number of vCPUs assigned to the virtual appliance

By default, the Workload Balancing virtual appliance is assigned 2 vCPUs. This value is sufficient for pools hosting 1000 VMs. You do not usually need to increase it. Only decrease the number of vCPUs assigned to the virtual appliance if you have a small environment.

This procedure explains how to change the number of vCPUs assigned to the Workload Balancing virtual appliance. Shut down the virtual appliance before performing these steps. Workload Balancing is unavailable for approximately five minutes.

1. Shut down the Workload Balancing virtual appliance.
2. In the XenCenter resource pane, select the Workload Balancing virtual appliance.
3. In the virtual appliance **General** tab, click **Properties**. The **Properties** dialog opens.
4. In the **CPU** tab of the **Properties** dialog, edit the CPU settings to the required values.
5. Click **OK**.
6. Start the Workload Balancing virtual appliance.

The new vCPU settings take affect when the virtual appliance starts.

Resize the virtual appliance memory

By default, the Workload Balancing virtual appliance is assigned 2 GB of memory.

For large pools, set the Workload Balancing virtual appliance to consume the maximum amount of memory you can make available to it (even up to 16 GB). Do not be concerned about high memory utilization. High memory utilization is normal for the virtual appliance because the database always consumes as much memory as it can obtain.

> **Note:**
>
> Dynamic Memory Control is not supported with the Workload Balancing virtual appliance. Set a fixed value for the maximum memory to assign to the virtual appliance.

This procedure explains how to resize the memory of the Workload Balancing virtual appliance. Shut down the virtual appliance before performing these steps. Workload Balancing is unavailable for approximately five minutes.

1. Shut down the Workload Balancing virtual appliance.
2. In the XenCenter resource pane, select the Workload Balancing virtual appliance.
3. In the virtual appliance **Memory** tab, click **Edit**. The **Memory Settings** dialog opens.
4. Edit the memory settings to the required values.
5. Click **OK**.
6. Start the Workload Balancing virtual appliance.

The new memory settings take affect when the virtual appliance starts.

Extend the virtual appliance disk

> **Warning:**
>
> You can only extend the available disk space in versions 8.3.0 and later as LVM is not supported before 8.3.0.
>
> Workload Balancing does not support decreasing the available disk space.

By default, the Workload Balancing virtual appliance is assigned 30 GB of disk space.

The greater the number of VMs the Workload Balancing virtual appliance is monitoring, the more disk space it consumes per day.

You can estimate the amount of disk size that the virtual appliance needs by using the following formula:

```
Total estimated disk size = ( ( number of days * average disk usage ) +
    base disk usage ) * grooming multiplier
```

- *number of days* is the number of days of data to retain
- *average disk usage* depends on the number of VMs being monitored. The following values give an approximation for certain numbers of VMs:
  - For 200 VMs - 0.246 GB/day
  - For 400 VMs - 0.505 GB/day
  - For 600 VMs - 0.724 GB/day
  - For 800 VMs - 0.887 GB/day
- *base disk usage* is 2.4 GB
- *grooming multiplier* is 1.25. This multiplier accounts for the amount of disk space required by grooming. It assumes that grooming requires an additional 25% of the total calculated disk space.

For versions 8.2.2 and earlier This procedure explains how to extend the virtual disk of the Workload Balancing virtual appliance for Workload Balancing versions 8.2.2 and earlier.

> **Warning:**
>
> We recommend taking a snapshot of your data before performing this procedure. Incorrectly performing these steps can result in corrupting the Workload Balancing virtual appliance.

1. Shut down the Workload Balancing virtual appliance.
2. In the XenCenter resource pane, select the Workload Balancing virtual appliance.
3. Click the **Storage** tab.
4. Select the `vdi_xvda` disk, and click the **Properties** button.
5. In the `vdi_xvda` **Properties** dialog, select **Size and Location**.
6. Increase the disk size as needed, and click **OK**.
7. Start the Workload Balancing virtual appliance and log in to it.
8. Run the following command on the Workload Balancing virtual appliance:

   ```
   resize2fs /dev/xvda
   <!--NeedCopy-->
   ```

9. Run the `df -h` command to confirm the new disk size.

Installing `resize2fs` If the `resize2fs` tool is not installed on the Workload Balancing virtual appliance, you can install it by using the following steps.

If you are connected to the internet, run the following command on the Workload Balancing virtual appliance:

```
yum install -y --enablerepo=base,updates --disablerepo=citrix-*
    e2fsprogs
<!--NeedCopy-->
```

If there is no internet access:

1. Download the following from https://centos.pkgs.org/7/centos-x86_64/.

   - `libss-1.42.9-7.el7.i686.rpm`
   - `e2fsprogs-libs-1.42.9-7.el7.x86_64.rpm`
   - `e2fsprogs-1.42.9-7.el7.x86_64.rpm`

2. Upload them to Workload Balancing VM using SCP or any other suitable tool.

3. Run the following command from Workload Balancing virtual appliance:

   ```
   rpm -ivh libss-*.rpm e2fsprogs-*.rpm
   <!--NeedCopy-->
   ```

   The tool `resize2fs` is now installed.

For versions 8.3.0 and later This procedure explains how to extend the virtual disk of the Workload Balancing virtual appliance for Workload Balancing versions 8.3.0 and later, using Linux Volume Manager (LVM).

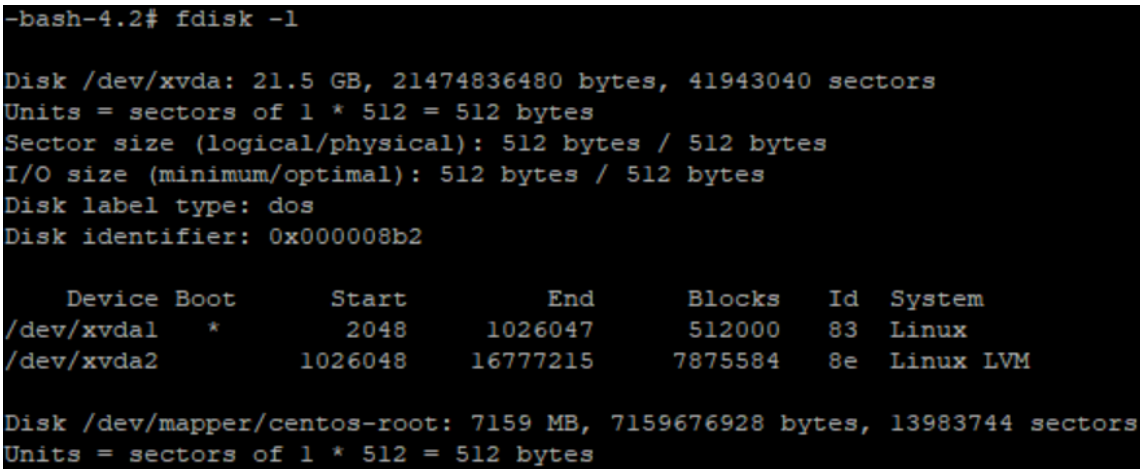
> **Warning:**
>
> This procedure must only be followed by Experienced Linux System Administrators as incorrectly performing these steps can result in corrupting the Workload Balancing virtual appliance. We cannot guarantee that problems resulting from the incorrect use of the Registry Editor can be solved. Be sure to back up the registry before you edit it and shut down the virtual appliance before performing these steps. Workload Balancing is unavailable for approximately five minutes.

To create new partitions, manipulate Physical Volumes and change your Filesystem size, perform the following actions while logged in as a Super User (root):

1. View the current partitions:

   ```
   fdisk -l
   <!--NeedCopy-->
   ```

The default partitions might look like this:



```
-bash-4.2# fdisk -l

Disk /dev/xvda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000008b2

    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1   *        2048     1026047      512000   83  Linux
/dev/xvda2         1026048    16777215     7875584   8e  Linux LVM

Disk /dev/mapper/centos-root: 7159 MB, 7159676928 bytes, 13983744 sectors
Units = sectors of 1 * 512 = 512 bytes
```

2. View the disk partition style:

```
parted <disk>
<!--NeedCopy-->
```

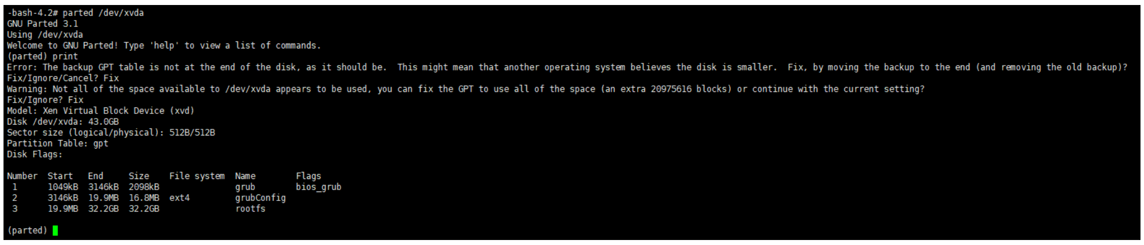
   For example, to view the partition style of /dev/xvda:

```
parted /dev/xvda
<!--NeedCopy-->
```

3. Enter p.

   If the following error messages occur, enter Fix to resolve each one:

   - "Error: The backup GPT table is not at the end of the disk, as it should be. This might mean that another operating system believes the disk is smaller. Fix, by moving the backup to the end (and removing the old backup)?"
   - "Warning: Not all of the space available to <disk> appears to be used, you can fix the GPT To use all of the space (an extra <block number> blocks) or continue with the current setting?"



```
-bash-4.2# parted /dev/xvda
GNU Parted 3.1
Using /dev/xvda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: The backup GPT table is not at the end of the disk, as it should be.  This might mean that another operating system believes the disk is smaller.  Fix, by moving the backup to the end (and removing the old backup)?
Fix/Ignore/Cancel? Fix
Warning: Not all of the space available to /dev/xvda appears to be used, you can fix the GPT to use all of the space (an extra 20975616 blocks) or continue with the current setting?
Fix/Ignore? Fix
Model: Xen Virtual Block Device (xvd)
Disk /dev/xvda: 43.0GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name        Flags
 1      1049kB  3146kB  2098kB               grub        bios_grub
 2      3146kB  19.9MB  16.8MB  ext4         grubConfig
 3      19.9MB  32.2GB  32.2GB               rootfs

(parted)
```

4. Enter q and press **Enter** to exit parted.

5. Edit the partitions:

```
fdisk <disk>
<!--NeedCopy-->
```

For example, to edit the partitions in Workload Balancing appliances:

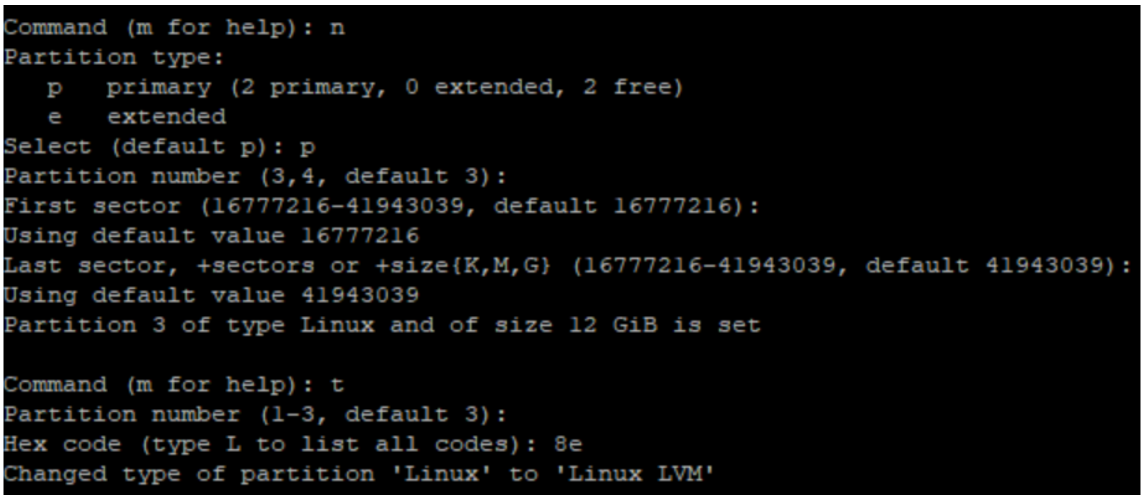
```
fdisk /dev/xvda
<!--NeedCopy-->
```

6. Type `n` and press **Enter** to create a new partition, type `p` and press **Enter** to make it a primary partition, and press **Enter** to use the default which is the next available partition (in this case, as stated above, it'll be partition number 3).

   > **Note:**
   >
   > If no additional space has been allocated yet, you will see a message indicating that there are no free sectors available. Type `q` and press **Enter** to quit fdisk. Allocate the desired space via XenCenter first and then come back to this step.

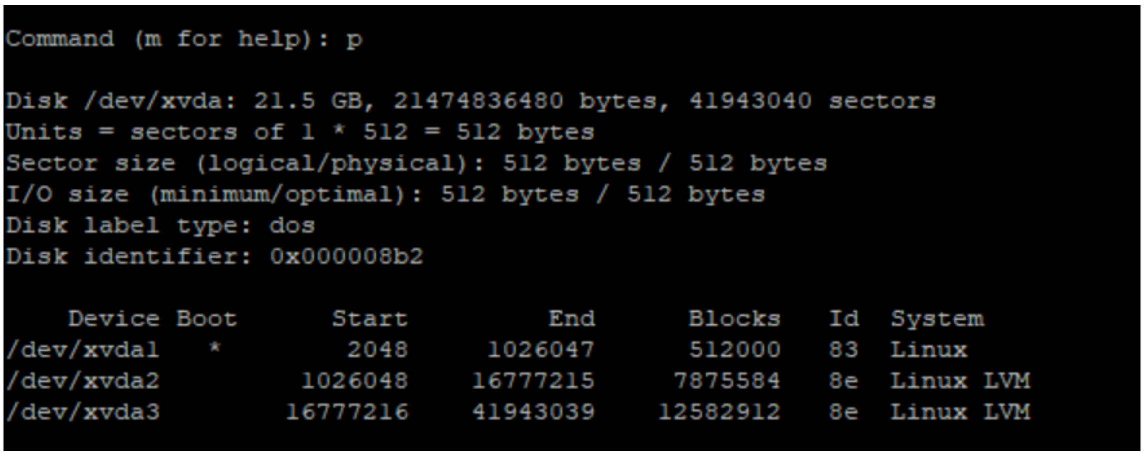
7. Press **Enter** twice to use the default first and last sectors of the available partition (or manually indicate the desired sectors). Type `t` to specify a partition type, choose the desired partition (in this case 3), type `8e`, and press **Enter** to make it an LVM type partition.

   Example output:



```
Command (m for help): n
Partition type:
   p   primary (2 primary, 0 extended, 2 free)
   e   extended
Select (default p): p
Partition number (3,4, default 3):
First sector (16777216-41943039, default 16777216):
Using default value 16777216
Last sector, +sectors or +size{K,M,G} (16777216-41943039, default 41943039):
Using default value 41943039
Partition 3 of type Linux and of size 12 GiB is set

Command (m for help): t
Partition number (1-3, default 3):
Hex code (type L to list all codes): 8e
Changed type of partition 'Linux' to 'Linux LVM'
```

8. Type `p` and press **Enter** to print the details of the partition. The output should look similar to the one below (note that the start and end blocks values might vary depending on the amount of space you've allocated):



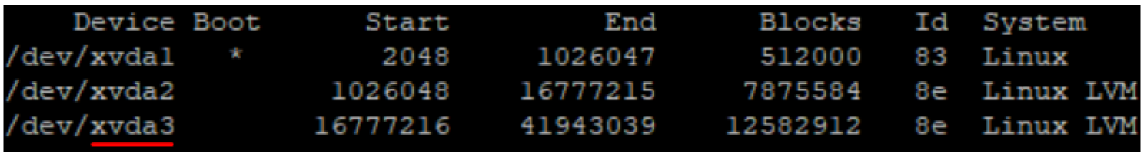
```
Command (m for help): p

Disk /dev/xvda: 21.5 GB, 21474836480 bytes, 41943040 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x000008b2

    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1   *        2048     1026047      512000   83  Linux
/dev/xvda2         1026048    16777215     7875584   8e  Linux LVM
/dev/xvda3        16777216    41943039    12582912   8e  Linux LVM
```

9. If something is incorrect, type `q` and press **Enter** to exit without saving and to prevent your existing partitions from being affected. Start again from step 1. Otherwise, if all looks well, type `w` and press **Enter** instead in order to write the changes.

   After writing these changes, you might get a warning indicating that the device was busy and the kernel is still using the old table. If that's the case, run this command which will refresh the partition table, before proceeding with the next step: `partprobe`.

   Make sure the new device partition (in this case `/dev/xvda4`) is listed now. To do so, run: `fdisk -l`.

   The newly created device should be listed now:



```
    Device Boot      Start         End      Blocks   Id  System
/dev/xvda1   *        2048     1026047      512000   83  Linux
/dev/xvda2         1026048    16777215     7875584   8e  Linux LVM
/dev/xvda3        16777216    41943039    12582912   8e  Linux LVM
```

10. If the output looks correct, create a Physical Volume:

```
pvcreate <new partition>
<!--NeedCopy-->
```

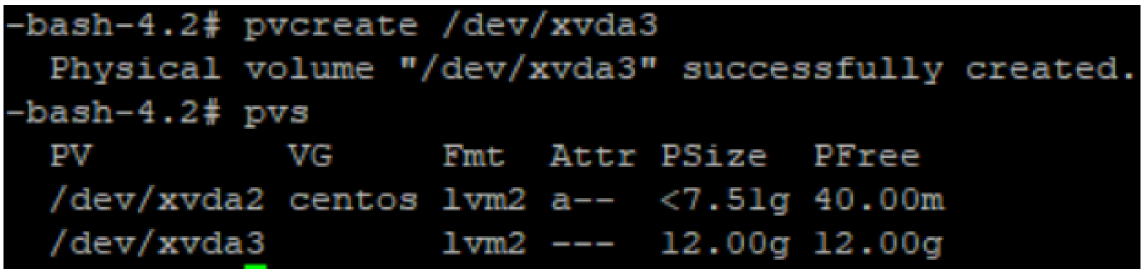
   For example:

```
pvcreate /dev/xvda4
<!--NeedCopy-->
```

11. Check that the Physical Volume created above is now listed:

```
pvs
<!--NeedCopy-->
```

   In this example, the additional space added was 12G. Example output:



```
-bash-4.2# pvcreate /dev/xvda3
  Physical volume "/dev/xvda3" successfully created.
-bash-4.2# pvs
  PV         VG     Fmt  Attr PSize  PFree
  /dev/xvda2 centos lvm2 a--  <7.51g 40.00m
  /dev/xvda3        lvm2 ---  12.00g 12.00g
```

12. Based on the output of the previous command, the Volume Group named centos must be extended:

    ```
    vgextend <volume group> <new partition>
    <!--NeedCopy-->
    ```

    For example:

    ```
    vgextend centos /dev/xvda4
    <!--NeedCopy-->
    ```

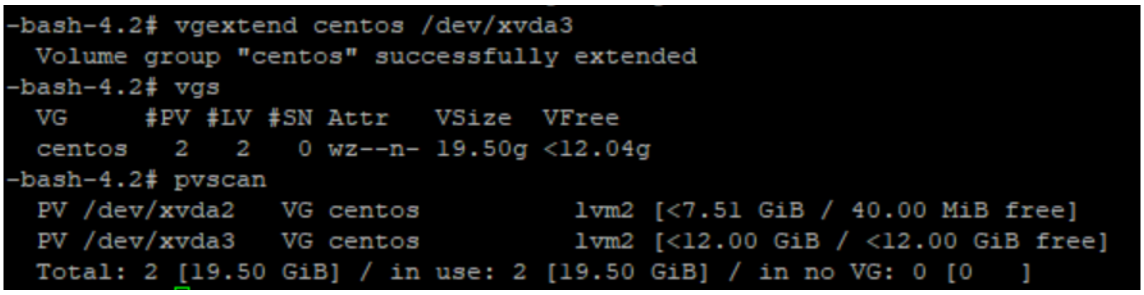
13. Check the current Volume Groups:

    ```
    vgs
    <!--NeedCopy-->
    ```

14. Run the following command:

    ```
    pvscan
    <!--NeedCopy-->
    ```

    This should show /dev/xvda4 as part of the centos Volume group. Example output:

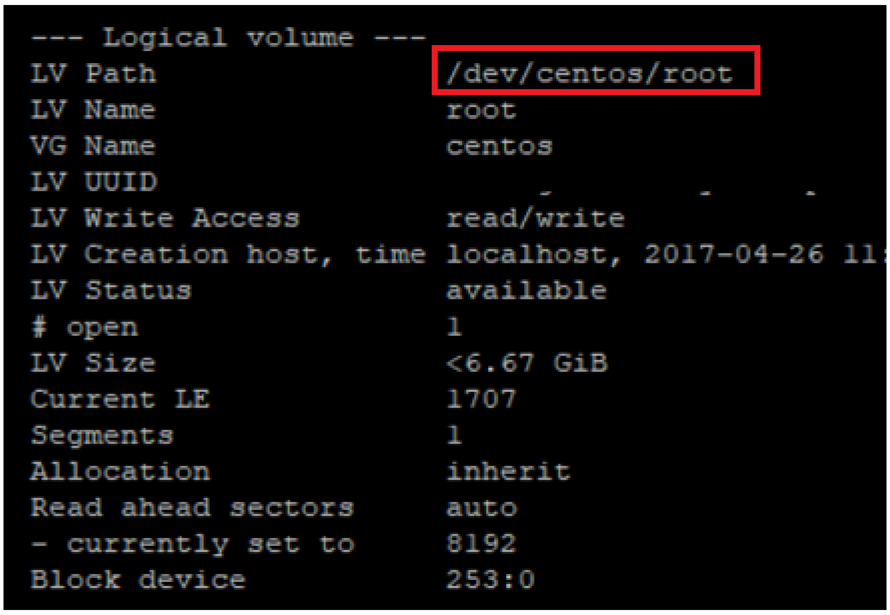


```
-bash-4.2# vgextend centos /dev/xvda3
  Volume group "centos" successfully extended
-bash-4.2# vgs
  VG     #PV #LV #SN Attr   VSize  VFree
  centos   2   2   0 wz--n- 19.50g <12.04g
-bash-4.2# pvscan
  PV /dev/xvda2   VG centos          lvm2 [<7.51 GiB / 40.00 MiB free]
  PV /dev/xvda3   VG centos          lvm2 [<12.00 GiB / <12.00 GiB free]
  Total: 2 [19.50 GiB] / in use: 2 [19.50 GiB] / in no VG: 0 [0   ]
```

15. If the information shown in the previous steps looks correct, run this command to see the Logic Volume path for the Logical Volume to be extended:

    ```
    lvdisplay
    <!--NeedCopy-->
    ```

    In this example, the path is /dev/centos/root:

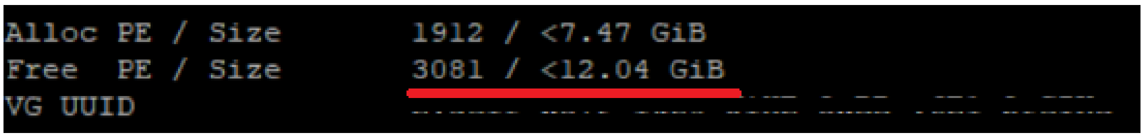


```
--- Logical volume ---
LV Path                /dev/centos/root
LV Name                root
VG Name                centos
LV UUID                [redacted]
LV Write Access        read/write
LV Creation host, time localhost, 2017-04-26 11:
LV Status              available
# open                 1
LV Size                <6.67 GiB
Current LE             1707
Segments               1
Allocation             inherit
Read ahead sectors     auto
- currently set to     8192
Block device           253:0
```

16. Run the following command to view the free PE/size (this tells you the exact value to use when extending the partition):

    ```
    vgdisplay
    <!--NeedCopy-->
    ```

    Example output:



```
Alloc PE / Size       1912 / <7.47 GiB
Free  PE / Size       3081 / <12.04 GiB
VG UUID               [redacted]
```

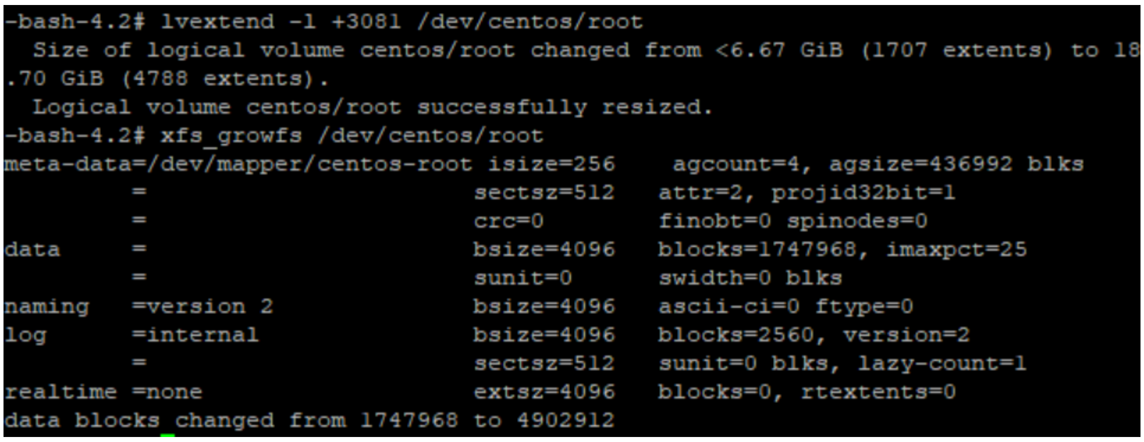
17. Using the free PE/size value and the Logic Volume path outputted in step 11, extend the Logic Volume:

    ```
    lvextend -l +100%FREE /dev/centos/root
    <!--NeedCopy-->
    ```

    If this executes successfully, extend the filesystem:

    ```
    resize2fs /dev/centos/root
    <!--NeedCopy-->
    ```

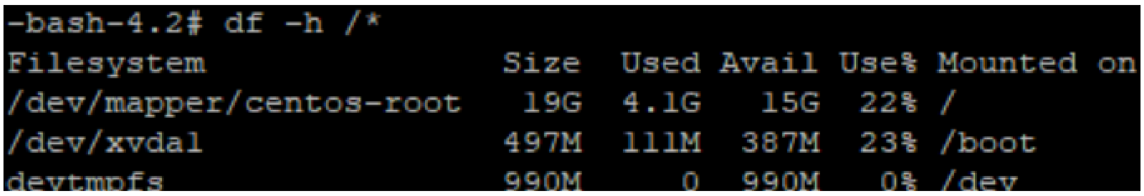
    Example output:



```
-bash-4.2# lvextend -l +3081 /dev/centos/root
  Size of logical volume centos/root changed from <6.67 GiB (1707 extents) to 18
.70 GiB (4788 extents).
  Logical volume centos/root successfully resized.
-bash-4.2# xfs_growfs /dev/centos/root
meta-data=/dev/mapper/centos-root isize=256    agcount=4, agsize=436992 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0 spinodes=0
data     =                       bsize=4096   blocks=1747968, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal               bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
data blocks changed from 1747968 to 4902912
```

18. Verify that the filesystem size shows as expected:

```
df -h /*
<!--NeedCopy-->
```



```
-bash-4.2# df -h /*
Filesystem               Size  Used Avail Use% Mounted on
/dev/mapper/centos-root   19G  4.1G   15G  22% /
/dev/xvda1               497M  111M  387M  23% /boot
devtmpfs                 990M     0  990M   0% /dev
```

If you're seeing the expected numbers, you have successfully allocated the desired space and correctly extended the partition. For further assistance, please contact XenServer Support.

Stop Workload Balancing

Because Workload Balancing is configured at the pool level, when you want it to stop managing a pool, you must do one of the following:

Pause Workload Balancing. Pausing Workload Balancing stops XenCenter from displaying recommendations for the specified resource pool and managing the pool. Pausing is designed for a short period and lets you resume monitoring without having to reconfigure. When you pause Workload Balancing, data collection stops for that resource pool until you enable Workload Balancing again.

1. In XenCenter, select the resource pool for which you want to disable Workload Balancing.
2. In the **WLB** tab, click **Pause**. A message appears on the **WLB** tab indicating that Workload Balancing is paused.

> **Tip:**
>
> To resume monitoring, click the **Resume** button in the **WLB** tab.

Disconnect the pool from Workload Balancing. Disconnecting from the Workload Balancing virtual appliance breaks the connection between the pool and if possible, deletes the pool data from the Workload Balancing database. When you disconnect from Workload Balancing, Workload Balancing stops collecting data on the pool.

1. In XenCenter, select the resource pool on which you want to stop Workload Balancing.
2. From the **Infrastructure** menu, select **Disconnect Workload Balancing Server**. The **Disconnect Workload Balancing server** dialog box appears.
3. Click **Disconnect** to stop Workload Balancing from monitoring the pool permanently.

> **Tip:**
>
> If you disconnected the pool from the Workload Balancing virtual appliance, to re-enable Workload Balancing on that pool, you must reconnect to a Workload Balancing appliance. For information, see the Connect to the Workload Balancing Virtual Appliance.

Enter maintenance mode with Workload Balancing enabled

With Workload Balancing enabled, if you put a host in maintenance mode, XenServer migrates the VMs running on that host to their optimal hosts when available. XenServer uses Workload Balancing recommendations that are based on performance data, your placement strategy, and performance thresholds to select the optimal host.

If an optimal host is not available, the words **Click here to suspend the VM** appear in the **Enter Maintenance Mode** wizard. In this case, because there is not a host with sufficient resources to run the VM, Workload Balancing does not recommend a placement. You can either suspend this VM or exit maintenance mode and suspend a VM on another host in the same pool. Then, if you reenter the **Enter Maintenance Mode** dialog box, Workload Balancing might be able to list a host that is a suitable candidate for migration.

> **Note:**
>
> When you take a host offline for maintenance and Workload Balancing is enabled, the words "Workload Balancing"appear in the **Enter Maintenance Mode** wizard.

To enter maintenance mode with Workload Balancing enabled:

1. In the **Resources** pane of XenCenter, select the physical server that you want to take off-line.
2. From the **Server** menu, select **Enter Maintenance Mode**.
3. In the **Enter Maintenance Mode** wizard, click **Enter maintenance mode**.

   The VMs running on the host are automatically migrated to the optimal host based on the Workload Balancing performance data, your placement strategy, and performance thresholds.

To take the server out of maintenance mode:

1. Right-click the host and select **Exit Maintenance Mode**.

   When you remove a host from maintenance mode, XenServer automatically restores that host's original VMs to that host.

Remove the Workload Balancing virtual appliance

To remove the Workload Balancing virtual appliance, we recommend you use the standard procedure to delete VMs from XenCenter.

When you delete the Workload Balancing virtual appliance, the PostgreSQL database containing the Workload Balancing is deleted. To save this data, you must migrate it from the database before deleting the Workload Balancing virtual appliance.

Manage the Workload Balancing database

The Workload Balancing database is a PostgreSQL database. PostgreSQL is an open-source relational database. You can find documentation for PostgreSQL by searching the web.

The following information is intended for database administrators and advanced users of PostgreSQL who are comfortable with database administration tasks. If you are not experienced with PostgreSQL, we recommend that you become familiar with it before you attempt the database tasks in the sections that follow.

By default, the PostgreSQL user name is `postgres`. You set the password for this account during Workload Balancing configuration.

The amount of historical data you can store is based on the size of the virtual disk allocated to Workload Balancing and the minimum required space. By default, the size of the virtual disk allocated to Workload Balancing is 30 GB. In terms of managing the database, you can control the space that database data consumes by configuring database grooming. For more information, see Database grooming parameters.

To store a lot of historical data, for example if you want to enable the Pool Audit trail Report, you can do either of the following:

- Make the virtual disk size assigned to the Workload Balancing virtual appliance larger. To do so, import the virtual appliance, and increase the size of the virtual disk by following the steps in Extend the virtual appliance disk.
- Create periodic duplicate backup copies of the data by enabling remote client access to the database and using a third-party database administration tool.

Access the database

The Workload Balancing virtual appliance has firewall configured in it. Before you can access the database, you must add the postgresQL server port to the iptables.

1. From the Workload Balancing virtual appliance console, run the following command:

   ```
   iptables -A INPUT -i eth0 -p tcp -m tcp --dport 5432 -m \
   state --state NEW,ESTABLISHED -j ACCEPT
   <!--NeedCopy-->
   ```

2. (Optional) To make this configuration persist after the virtual appliance is rebooted, run the following command:

   ```
   iptables-save > /etc/sysconfig/potables
   <!--NeedCopy-->
   ```

Control database grooming

The Workload Balancing database automatically deletes the oldest data whenever the virtual appliance reaches the minimum amount of disk space that Workload Balancing requires to run. By default, the minimum amount of required disk space is set to 1,024 MB.

The Workload Balancing database grooming options are controlled through the file `wlb.conf`.

When there is not enough disk space left on the Workload Balancing virtual appliance, Workload Balancing automatically starts grooming historical data. The process is as follows:

1. At a predefined grooming interval, the Workload Balancing data collector checks if grooming is required. Grooming is required if the database data has grown to the point where the only space that remains unused is the minimum required disk space. Use `GroomingRequiredMinimumDiskSizeInMB` to set the minimum required disk space.

   You can change the grooming interval if desired using `GroomingIntervalInHour`. However, by default Workload Balancing checks to see if grooming is required once per hour.

2. If grooming is required, Workload Balancing begins by grooming the data from the oldest day. Workload Balancing then checks to see if there is now enough disk space for it to meet the minimum disk-space requirement.

3. If the first grooming did not free enough disk space, then Workload Balancing repeats grooming up to `GroomingRetryCounter` times without waiting for `GroomingIntervalInHour` hour.

4. If the first or repeated grooming freed enough disk space, then Workload Balancing waits for `GroomingIntervalInHour` hour and returns to Step 1.

5. If the grooming initiated by the `GroomingRetryCounter` did not free enough disk space, then Workload Balancing waits for `GroomingIntervalInHour` hour and returns to Step 1.

Database grooming parameters

There are five parameters in the `wlb.conf` file that control various aspects of database grooming. They are as follows:

- `GroomingIntervalInHour`. Controls how many hours elapse before the next grooming check is done. For example, if you enter **1**, Workload Balancing checks the disk space hourly. If you enter **2**, Workload Balancing checks disk space every two hours to determine if grooming must occur.
- `GroomingRetryCounter`. Controls the number of times Workload balancing tries rerunning the grooming database query.
- `GroomingDBDataTrimDays`. Controls the number of days worth of data Workload Balancing deletes from the database each time it tries to groom data. The default value is one day.
- `GroomingDBTimeoutInMinute`. Controls the number of minutes that the database grooming takes before it times out and is canceled. If the grooming query takes longer than is expected and does not finish running within the timeout period, the grooming task is canceled. The default value is 0 minutes, which means that database grooming never times out.
- `GroomingRequiredMinimumDiskSizeInMB`. Controls the minimum amount of free space left in the virtual disk assigned to the Workload Balancing virtual appliance. When the data in the virtual disk grows until there is only minimum disk size left on the virtual disk, Workload Balancing triggers database grooming. The default value is 2,048 MB.

To edit these values, see Edit the Workload Balancing configuration file.

Change the database password

We recommend using the `wlbconfig` command to change the database password. For more information, see Modify the Workload Balancing configuration options. Do not change the password by modifying the `wlb.conf` file.

Archive database data

To avoid having older historical data deleted, you can, optionally, copy data from the database for archiving. To do so, you must perform the following tasks:

1. Enable client authentication on the database.

2. Set up archiving using the PostgreSQL database administration tool of your choice.

Enable client authentication to the database

While you can connect directly to the database through the Workload Balancing console, you can also use a PostgreSQL database management tool. After downloading a database management tool, install it on the system from which you want to connect to the database. For example, you can install the tool on the same laptop where you run XenCenter.

Before you can enable remote client authentication to the database, you must:

1. Modify the database configuration files, including pg_hba.conf file and the postgresql.conf, to allow connections.
2. Stop the Workload Balancing services, restart the database, and then restart the Workload Balancing services.
3. In the database management tool, configure the IP address of the database (that is, the IP address of the Workload Balancing virtual appliance) and the database password.

Modify the database configuration files

To enable client authentication on the database, you must modify the following files on the Workload Balancing virtual appliance: the `pg_hba.conf` file and the `postgresql.conf` file.

To edit the `pg_hba.conf` file:

1. Modify the `pg_hba.conf` file. From the Workload Balancing virtual appliance console, open the `pg_hba.conf` file with an editor, such as VI. For example:

   ```
   vi /var/lib/pgsql/9.0/data/pg_hba.conf
   <!--NeedCopy-->
   ```

2. If your network uses IPv4, add the IP address from the connecting computer to this file. For example:

   In the configuration section, enter the following under `#IPv4 local connections`:

   - **TYPE:** host
   - **DATABASE:** all
   - **USER:** all
   - **CIDR-ADDRESS:** 0.0.0.0/0
   - **METHOD:** trust

3. Enter your IP address in the `CIDR-ADDRESS` field.

> **Note:**
>
> Instead of entering 0.0.0.0/0, you can enter your IP address and replace the last three digits with 0/24. The trailing "24"after the / defines the subnet mask and only allows connections from IP addresses within that subnet mask.

When you enter `trust` for the `Method` field, it enables the connection to authenticate without requiring a password. If you enter `password` for the `Method` field, you must supply a password when connecting to the database.

4. If your network uses IPv6, add the IP address from the connecting computer to this file. For example:

   Enter the following under #`IPv6 local connections`:

   - **TYPE:** host
   - **DATABASE:** all
   - **USER:** all
   - **CIDR-ADDRESS:** `::0/0`
   - **METHOD:** trust

   Enter the IPv6 addresses in the `CIDR-ADDRESS` field. In this example, the `::0/0` opens the database up to connections from any IPv6 addresses.

5. Save the file and quit the editor.

6. After changing any database configurations, you must restart the database to apply the changes. Run the following command:

   ```
   service postgresql-9.0 restart
   <!--NeedCopy-->
   ```

To edit the `postgresql.conf` file:

1. Modify the `postgresql.conf` file. From the Workload Balancing virtual appliance console, open the `postgresql.conf` file with an editor, such as VI. For example:

   ```
   vi /var/lib/pgsql/9.0/data/postgresql.conf
   <!--NeedCopy-->
   ```

2. Edit the file so that it listens on any port and not just the local host. For example:

   a) Find the following line:

   ```
   # listen_addresses='localhost'
   <!--NeedCopy-->
   ```

   b) Remove the comment symbol (#) and edit the line to read as follows:

```
listen_addresses='*'
<!--NeedCopy-->
```

3. Save the file and quit the editor.
4. After changing any database configurations, you must restart the database to apply the changes. Run the following command:

   ```
   service postgresql-9.0 restart
   <!--NeedCopy-->
   ```

Change the database maintenance window

Workload Balancing automatically performs routine database maintenance daily at 12:05AM GMT (00:05), by default. During this maintenance window, data collection occurs but the recording of data might be delayed. However, the Workload Balancing user interface controls are available during this period and Workload Balancing still makes optimization recommendations.

> **Note:**
>
> To avoid a loss of Workload Balancing:
>
> - During the maintenance window, the Workload Balancing server restarts. Ensure that you do not restart your VMs at the same time.
> - At other times, when restarting all VMs in your pool, do not restart the Workload Balancing server.

Database maintenance includes releasing allocated unused disk space and reindexing the database. Maintenance lasts for approximately 6 to 8 minutes. In larger pools, maintenance might last longer, depending on how long Workload Balancing takes to perform discovery.

Depending on your time zone, you might want to change the time when maintenance occurs. For example, in the Japan Standard Time (JST) time zone, Workload Balancing maintenance occurs at 9:05 AM (09:05), which can conflict with peak usage in some organizations. If you want to specify a seasonal time change, such as Daylight Saving Time or summer time, you must build the change into value you enter.

To change the maintenance time:

1. In the Workload Balancing console, run the following command from any directory:

   ```
   crontab -e
   <!--NeedCopy-->
   ```

   Workload Balancing displays the following:

```
05 0 * * * /opt/vpx/wlb/wlbmaintenance.sh
<!--NeedCopy-->
```

The value `05 0` represents the default time for Workload Balancing to perform maintenance in minutes (05) and then hours (0). (The asterisks represent the day, month, and year the job runs: Do not edit these fields.) The entry `05 0` indicates that database maintenance occurs at 12:05 AM, or 00:05, Greenwich Mean Time (GMT) every night. This setting means that if you live in New York, the maintenance runs at 7:05 PM (19:05) during winter months and 8:05 PM in summer months.

> **Important:**
>
> Do not edit the day, month, and year the job runs (as represented by asterisks). Database maintenance must run daily.

2. Enter the time at which you want maintenance to occur in GMT.
3. Save the file and quit the editor.

Customize Workload Balancing

Workload Balancing provides several methods of customization:

- **Command lines for scripting**. For more information, see Workload Balancing commands.
- **Host Power On scripting support**. You can also customize Workload Balancing (indirectly) through the Host Power On scripting. For more information, see Hosts and resource pools.

Upgrade Workload Balancing

Online upgrading of Workload Balancing has been deprecated for security reasons. Customers cannot upgrade by using the yum repo anymore. Customers can upgrade Workload Balancing to the latest version by importing the latest Workload Balancing virtual appliance downloadable at the XenServer Downloads page.

Workload Balancing commands

This section provides a reference for the Workload Balancing commands. You can perform these commands from the XenServer host or console to control Workload Balancing or configure Workload Balancing settings on the XenServer host. This appendix includes xe commands and 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.

Log 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 extra 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 login prompt, enter the account user name.
2. At the Password prompt, enter the password for the account:

   > **Note:**
   >
   > To log off from the Workload Balancing virtual appliance, simply type `logout` at the command prompt.

`wlb restart`

Run the `wlb restart` command from anywhere in the Workload Balancing appliance to stop and then restart the Workload Balancing Data Collection, Web Service, and Data Analysis services.

`wlb start`

Run the `wlb start` command from anywhere in the Workload Balancing appliance to start the Workload Balancing Data Collection, Web Service, and Data Analysis services.

`wlb stop`

Run the `wlb stop` command from anywhere in the Workload Balancing appliance to stop the Workload Balancing Data Collection, Web Service, and Data Analysis services.

`wlb status`

Run the `wlb status` command from anywhere in the Workload Balancing appliance to determine the status of the Workload Balancing server. After you run this command, the status of the three Workload Balancing services (the Web Service, Data Collection Service, and Data Analysis Service) is displayed.

Modify 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. The `wlb.conf` file is a configuration file on the Workload Balancing virtual appliance.

To modify the most commonly used options, use the command `wlb config`. Running the `wlb config` 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 run this command, the Workload Balancing services are restarted.

Run the following command on the Workload Balancing virtual appliance:

```
wlb config
<!--NeedCopy-->
```

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. Therefore, if the configuration parameters you specify are not within the required range, Workload Balancing does not generate an error log.

Edit the Workload Balancing configuration file

You can modify Workload Balancing configuration options by editing the `wlb.conf` file, which is stored in `/opt/vpx/wlb` directory on the Workload Balancing virtual appliance. In general, only change the settings in this file with guidance from XenServer. However, there are three categories of settings you can change if desired:

- **Workload Balancing account name and password**. It is easier to modify these credentials by running the `wlb config` command.
- **Database password**. This value can be modified using the wlb.conf file. However, we recommend modifying it through the `wlb config` 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. However, if you do so, we recommend using caution.

For all other settings in the `wlb.conf` file, we recommend leaving them at their default, unless you have been instructed 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/vpx/wlb/wlb.conf
   <!--NeedCopy-->
   ```

   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. Therefore, if the configuration parameters you specify are not within the required range, Workload Balancing does not generate an error log.

Increase 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/vpx/wlb/wlb.conf`. Enter a 1 or true to enable logging for a specific trace and a 0 or false to disable logging. For example, to enable logging for the Analysis Engine trace, enter:

```
AnalEngTrace=1
<!--NeedCopy-->
```

You might want to increase logging detail before reporting an issue to XenServer 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 increases the log file size quickly.
Data Collection Trace	`DataCollectionTrace`	Logs the actions of retrieving metrics. This value lets you see the metrics Workload Balancing is retrieving and inserting into the Workload Balancing data store. However, leaving this trace on increases 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 increases the log-file size quickly.
Queue Management Trace	`QueueManagementTrace`	Logs details about data collection queue management processing. (This option 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 server 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 hosts for VM placement.
Audit Log Trace	`AuditLogTrace`	Shows the action of the audit log data being captured and written. (This option is only for internal use and does not provide information that is captured in the audit log.) However, leaving this trace on increases the log file size quickly.
Scheduled Task Trace	`ScheduledTaskTrace`	Logs details about scheduled tasks. For example, if your scheduled mode changes 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.

Certificates for Workload Balancing

August 24, 2023

XenServer and Workload Balancing communicate over HTTPS. During Workload Balancing Configuration, the wizard automatically creates a self-signed test certificate. This self-signed test certificate lets Workload Balancing establish a TLS connection to XenServer. By default, Workload Balancing creates this TLS connection with XenServer automatically. You do not need to perform any certificate configurations during or after configuration for Workload Balancing to create this TLS connection.

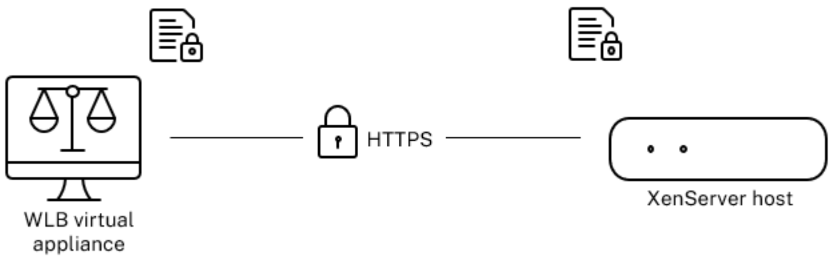
> **Note:**
>
> The self-signed certificate is a placeholder to facilitate HTTPS communication and is not from a trusted certificate authority. For added security, we recommend using a certificate signed from a trusted certificate authority.

To use a certificate from another certificate authority, such as a signed one from a commercial authority, you must configure Workload Balancing and XenServer to use it.

By default, XenServer does not validate the identity of the certificate before it establishes connection to Workload Balancing. To configure XenServer to check for a specific certificate, export the root certificate that was used to sign the certificate. Copy the certificate to XenServer and configure XenServer to check for it when a connection to Workload Balancing is made. XenServer acts as the client in this scenario and Workload Balancing acts as the server.

Depending on your security goals, you can either:

- Configure XenServer to verify the self-signed certificate.
- Configure XenServer to verify a certificate-authority certificate.



> **Note:**
>
> Certificate verification is a security measure designed to prevent unwanted connections. Workload Balancing certificates must meet strict requirements or the certificate verification doesn't succeed. When certificate verification fails, XenServer doesn't allow the connection.
>
> For certificate verification to succeed, you must store the certificates in the specific locations in which XenServer expects to find the certificates.

Configure XenServer to verify the self-signed certificate

You can configure XenServer to verify that the XenServer Workload Balancing self-signed certificate is authentic before XenServer permits Workload Balancing to connect.

> **Important:**
>
> To verify the XenServer Workload Balancing self-signed certificate, you must connect to Workload Balancing using its host name. To find the Workload Balancing host name, run the `hostname` command on the virtual appliance.

To configure XenServer to verify the self-signed certificate, complete the following steps:

1. Copy the self-signed certificate from the Workload Balancing virtual appliance to the pool coordinator. The XenServer Workload Balancing self-signed certificate is stored at `/etc/ssl/certs/server.pem`. Run the following command on the pool coordinator:

   ```
   scp root@<wlb-ip>:/etc/ssl/certs/server.pem .
   <!--NeedCopy-->
   ```

2. If you receive a message stating that the authenticity of `wlb-ip` cannot be established, type `yes` to continue.
3. Enter Workload Balancing virtual appliance root password when prompted. The certificate is copied to the current directory.
4. Install the certificate. Run the following command in the directory where you copied the certificate:

   ```
   xe pool-certificate-install filename=server.pem
   <!--NeedCopy-->
   ```

5. Verify the certificate was installed correctly by running the following command on the pool coordinator:

   ```
   xe pool-certificate-list
   <!--NeedCopy-->
   ```

   If you installed the certificate correctly, the output of this command includes the exported root certificate. Running this command lists all installed TLS certificates, including the certificate you installed.

6. To synchronize the certificate from the coordinator to all hosts in the pool, running the following command on the pool coordinator:

   ```
   xe pool-certificate-sync
   <!--NeedCopy-->
   ```

   Running the `pool-certificate-sync` command on the coordinator synchronizes the certificate and certificate revocation lists on all the pool hosts with the coordinator. This action ensures all hosts in the pool use the same certificates.

   There is no output from this command. However, the next step does not work if this one did not work successfully.

7. Instruct XenServer to verify the certificate before connecting to the Workload Balancing virtual appliance. Run the following command on the pool coordinator:

   ```
   xe pool-param-set wlb-verify-cert=true uuid=uuid_of_pool
   <!--NeedCopy-->
   ```

   > **Tip:**
   >
   > Pressing the **Tab** key automatically populates the UUID of the pool.

8. (Optional) To verify this procedure worked successfully, perform the following steps:

   a) To test if the certificate synchronized to the other hosts in the pool, run the `pool-certificate-list` command on those hosts.

   b) To test if XenServer was set to verify the certificate, run the `pool-param-get` command with the `param-name`=wlb-verify-cert parameter. For example:

   ```
   xe pool-param-get param-name=wlb-verify-cert uuid=uuid_of_pool
   <!--NeedCopy-->
   ```

Configure XenServer to verify a certificate-authority certificate

You can configure XenServer to verify a certificate signed by a trusted certificate authority.

For trusted authority certificates, XenServer requires an exported certificate or certificate chain (the intermediate and root certificates) in `.pem` format that contains the public key.

If you want Workload Balancing to use a trusted authority certificate, do the following tasks:

1. Obtain a signed certificate from the certificate authority.
2. Specify and apply the new certificate.
3. Import the certificate chain into the pool.

Before beginning these tasks, ensure:

- You know the IP address for the XenServer pool coordinator.
- XenServer can resolve the Workload Balancing host name. (For example, you can try pinging the Workload Balancing FQDN from the XenServer console for the pool coordinator.)

Obtain a signed certificate from the certificate authority

To obtain a certificate from a certificate authority, you must generate a Certificate Signing Request (CSR). On the Workload Balancing virtual appliance, create a private key and use that private key to generate the CSR.

Guidelines for specifying the Common Name The Common Name (CN) you specify when creating a CSR must exactly match the FQDN of your Workload Balancing virtual appliance. It must also match the FQDN or IP address you specified in the **Address** box in the **Connect to WLB Server** dialog box.

To ensure the name matches, specify the Common Name using one of these guidelines:

- Specify the same information for the certificate's Common Name as you specified in the **Connect to WLB Server** dialog.

  For example, if your Workload Balancing virtual appliance is named `wlb-vpx.yourdomain`, specify `wlb-vpx.yourdomain` in the **Connect to WLB Server** dialog and provide `wlb-vpx.yourdomain` as the Common Name when creating the CSR.

- If you connected your pool to Workload Balancing by IP address, use the FQDN as the Common Name and the IP address as a Subject Alternative Name (SAN). However, this approach might not work in all situations.

Create a private key file On the Workload Balancing virtual appliance, complete the following steps:

1. Create a private key file:

   ```
   openssl genrsa -des3 -out privatekey.pem 2048
   <!--NeedCopy-->
   ```

2. Remove the password:

   ```
   openssl rsa -in privatekey.pem -out privatekey.nop.pem
   <!--NeedCopy-->
   ```

> **Note:**
>
> If you enter the password incorrectly or inconsistently, you might receive some messages indicating that there is a user interface error. You can ignore the message and rerun the command to create the private key file.

Generate the Certificate Signing Request On the Workload Balancing virtual appliance, complete the following steps:

1. Create the Certificate Signing Request (CSR) using the private key:

   ```
   openssl req -new -key privatekey.nop.pem -out csr
   <!--NeedCopy-->
   ```

2. Follow the prompts to provide the information necessary to generate the CSR:

Country Name. Enter the TLS Certificate country codes for your country. For example, CA for Canada or JM for Jamaica. You can find a list of TLS Certificate country codes on the web.

State or Province Name (full name). Enter the state or province where the pool is located. For example, Massachusetts or Alberta.

Locality Name. The name of the city where the pool is located.

Organization Name. The name of your company or organization.

Organizational Unit Name. Enter the department name. This field is optional.

Common Name. Enter the FQDN of your Workload Balancing server. This value must match the name the pool uses to connect to Workload Balancing. For more information, see Guidelines for specifying the Common Name.

Email Address. This email address is included in the certificate when you generate it.

3. Provide optional attributes or click Enter to skip providing this information.

   The CSR request is saved in the current directory and is named `csr`.

4. Display the CSR in the console window by running the following commands in the Workload Balancing appliance console:

```
cat csr
<!--NeedCopy-->
```

5. Copy the entire CSR and use it to request the certificate from the certificate authority.

Specify and apply the new certificate

Use this procedure to specify Workload Balancing use a certificate from a certificate authority. This procedure installs the root and (if available) intermediate certificates.

To specify a new certificate, complete the following steps:

1. Download the signed certificate, root certificate and, if the certificate authority has one, the intermediate certificate from the certificate authority.

2. If you didn't download the certificates directly to the Workload Balancing virtual appliance, copy them across by using one of the following methods:

   - From a Windows computer, use WinSCP or another copying utility.

     For the host name, you can enter the IP address and leave the port at the default. The user name and password are typically root and whatever password you set during configuration.

- From a Linux computer to the Workload Balancing appliance, use SCP or another copying utility. For example:

  ```
  scp root_ca.pem root@wlb-ip:/path_on_your_WLB
  <!--NeedCopy-->
  ```

3. On the Workload Balancing virtual appliance, merge the contents of all the certificates (root certificate, intermediate certificate - if it exists, and signed certificate) into one file. You can use the following command:

   ```
   cat signed_cert.pem intermediate_ca.pem root_ca.pem > server.pem
   <!--NeedCopy-->
   ```

4. Rename the existing certificate and key by using the move command:

   ```
   mv /etc/ssl/certs/server.pem /etc/ssl/certs/server.pem_orig
   mv /etc/ssl/certs/server.key /etc/ssl/certs/server.key_orig
   <!--NeedCopy-->
   ```

5. Copy the merged certificate:

   ```
   mv server.pem /etc/ssl/certs/server.pem
   <!--NeedCopy-->
   ```

6. Copy the private key created previously:

   ```
   mv privatekey.nop.pem /etc/ssl/certs/server.key
   <!--NeedCopy-->
   ```

7. Make the private key readable only by root. Use the `chmod` command to fix permissions.

   ```
   chmod 600 /etc/ssl/certs/server.key
   <!--NeedCopy-->
   ```

8. Restart `stunnel`:

   ```
   killall stunnel
   stunnel
   <!--NeedCopy-->
   ```

Import the certificate chain into the pool

After you obtain the certificates, import them onto the XenServer pool coordinator. Synchronize the hosts in the pool to use those certificates. Then you can configure XenServer to check the certificate identity and validity each time Workload Balancing connects to a host.

1. Copy the signed certificate, root certificate and, if the certificate authority has one, the intermediate certificate from the certificate authority onto the XenServer pool coordinator.

2. Install the root certificate on the pool coordinator:

   ```
   xe pool-install-ca-certificate filename=root_ca.pem
   <!--NeedCopy-->
   ```

3. If applicable, install the intermediate certificate on the pool coordinator:

   ```
   xe pool-install-ca-certificate filename=intermediate_ca.pem
   <!--NeedCopy-->
   ```

4. Verify both the certificates installed correctly by running this command on the pool coordinator:

   ```
   xe pool-certificate-list
   <!--NeedCopy-->
   ```

   Running this command lists all installed TLS certificates. If the certificates installed successfully, they appear in this list.

5. Synchronize the certificate on the pool coordinator to all hosts in the pool:

   ```
   xe pool-certificate-sync
   <!--NeedCopy-->
   ```

   Running the `pool-certificate-sync` command on the coordinator synchronizes the certificates and certificate revocation lists on all the pool hosts with the pool coordinator. This action ensures all hosts in the pool use the same certificates.

6. Instruct XenServer to verify a certificate before connecting to the Workload Balancing virtual appliance. Run the following command on the pool coordinator:

   ```
   xe pool-param-set wlb-verify-cert=true uuid=uuid_of_pool
   <!--NeedCopy-->
   ```

   > **Tip:**
   >
   > Pressing the Tab key automatically populates the UUID of the pool.

7. If you specified an IP address in the **Connect to WLB** dialog before you enabled certificate verification, you might be prompted to reconnect the pool to Workload Balancing.

   Specify the FQDN for the Workload Balancing appliance in **Address** in the **Connect to WLB** dialog exactly as it appears in the certificate's Common Name. Enter the FQDN to ensure that the Common Name matches the name that XenServer uses to connect.

Troubleshooting

- If the pool cannot connect to Workload Balancing after configuring certificate verification, check to see if the pool can connect if you turn certificate verification off. You can use the command `xe` `pool-param-set wlb-verify-cert=false uuid=uuid_of_pool` to disable certificate verification. If it can connect with verification off, the issue is in your certificate configuration. If it cannot connect, the issue is in either your Workload Balancing credentials or your network connection.

- Some commercial certificate authorities provide tools to verify the certificate installed correctly. Consider running these tools if these procedures fail to help isolate the issue. If these tools require specifying a TLS port, specify port 8012 or whatever port you set during Workload Balancing Configuration.
- If the **WLB** tab shows a connection error, there might be a conflict between the certificate Common Name and the name of the Workload Balancing virtual appliance. The Workload Balancing virtual appliance name and the Common Name of the certificate must match exactly.

For more information, see Troubleshooting.

Troubleshoot Workload Balancing

March 14, 2024

While Workload Balancing usually runs smoothly, this series of sections provides guidance in case you encounter issues.

> **Notes:**
>
> - Workload Balancing is available for XenServer Premium Edition customers. For more information about XenServer licensing, see Licensing. To upgrade, or to buy a XenServer license, visit the XenServer website.
> - Workload Balancing 8.3.0 and later are compatible with XenServer 8 and Citrix Hypervisor 8.2 Cumulative Update 1.

Determine the status of the Workload Balancing virtual appliance

Run the `systemctl status workloadbalancing` command. For more information, see Workload Balancing commands.

General troubleshooting tips

- Start troubleshooting by reviewing the Workload Balancing log files (`LogFile.log` and `wlb_install_log.log`). You can find these logs in Workload Balancing virtual appliance in this location (by default):

`/var/log/wlb`

The level of detail in these log files can be configured by using the `wlb.conf` file. For more information, see Increase the detail in the Workload Balancing log.

- Check the logs in the XenCenter **Logs** tab for further information.
- To check the Workload Balancing virtual appliance build number, run the following command on a host in a pool that the virtual appliance monitors:

```
xe pool-retrieve-wlb-diagnostics | more
<!--NeedCopy-->
```

  The Workload Balancing version number appears at the top of the output.

- The Workload Balancing virtual appliance is based on the CentOS operating system. If you experience CPU, memory, or disk related issues in the virtual appliance, you can use the standard Linux logs in `/var/log/*` to analyse the issue.
- Use standard Linux debugging and performance tuning commands to understand the virtual appliance behavior. For example, `top`, `ps`, `free`, `sar`, and `netstat`.

Error messages

Workload Balancing displays errors on screen as dialog boxes and as error messages in the **Logs** tab in XenCenter.

If an error message appears, review the XenCenter event log for additional information. For more information, see the XenCenter product documentation.

Issues entering Workload Balancing credentials

If you cannot successfully enter the virtual appliance user account and password while configuring the **Connect to WLB Server** dialog, try the following:

- Ensure that Workload Balancing virtual appliance imported and was configured correctly and all of its services are running.
- Check to ensure that you are entering the correct credentials. The **Connect to WLB Server** dialog asks for two different credentials:
  - **WLB Server Credentials**: XenServer uses this account to communicate with Workload Balancing. You created this account on the Workload Balancing virtual appliance during Workload Balancing Configuration. By default, the user name for this account is `wlbuser`
    .

  - **Citrix Hypervisor Credentials**: This account is used by the Workload Balancing virtual appliance to connect to the XenServer pool. This account is created on the XenServer pool coordinator and has the `pool-admin` or `pool-operator` role.

- You can enter a host name in the **Address** box, but it must be the fully qualified domain name (FQDN) of the Workload Balancing virtual appliance. Do not enter the host name of the physical server hosting the appliance. If you are having trouble entering a computer name, try using the Workload Balancing appliance's IP address instead.
- Verify that the host is using the correct DNS server and the XenServer host can contact Workload Balancing server using its FQDN. To do this check, ping the Workload Balancing appliance using its FQDN from the XenServer host. For example, enter the following in the XenServer host console:

```
ping wlb-vpx-1.mydomain.net
<!--NeedCopy-->
```

Issues with firewalls

The following error appears if the Workload Balancing virtual appliance is behind a hardware firewall, and you did not configure the appropriate firewall settings: "There was an error connecting to the Workload Balancing server: <pool name> Click **Initialize WLB** to reinitialize the connection settings." This error might also appear if the Workload Balancing appliance is otherwise unreachable.

If the Workload Balancing virtual appliance is behind a firewall, open port 8012.

Likewise, the port XenServer uses to contact Workload Balancing (8012 by default), must match the port number specified when you ran the Workload Balancing Configuration wizard.

Workload Balancing connection errors

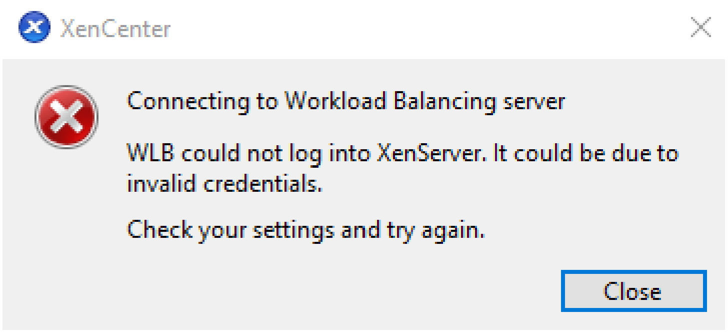
If you receive a connection error after configuring and connecting to Workload Balancing, the credentials might no longer be valid. To isolate this issue:

1. Verify that the credentials you entered in the **Connect to WLB Server** dialog box are correct. For more information, see scenario 1 and 2.
2. Verify that the IP address or FQDN for the Workload Balancing virtual appliance that you entered in the **Connect to WLB Server** dialog box is correct.
3. Verify that the user name you created during Workload Balancing configuration matches the credentials you entered in the **Connect to WLB Server** dialog box.

4. If you receive a connection error in the Workload Balancing Status line on the **WLB** tab, you might need to reconfigure Workload Balancing on that pool. Click the **Connect** button on the **WLB** tab and reenter the host credentials.

You may encounter one of the following scenarios when attempting to establish a connection from XenCenter to the Workload Balancing virtual appliance.

Scenario 1


XenCenter

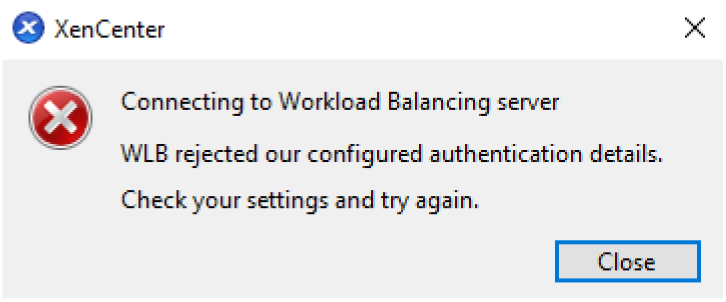
Connecting to Workload Balancing server

WLB could not log into XenServer. It could be due to invalid credentials.

Check your settings and try again.

This means that the credentials entered in the **Citrix Hypervisor Credentials** field in the **Connect to WLB Server** dialog box are incorrect. To fix this, double-check the credentials or check the **Use the current XenCenter credentials** box.

Scenario 2


XenCenter

Connecting to Workload Balancing server

WLB rejected our configured authentication details.

Check your settings and try again.

This means that there is a problem with the credentials entered in the **WLB Server Credentials** field in the **Connect to WLB Server** dialog box when attempting to connect to the Workload Balancing virtual appliance (either the username or the password are incorrect). However, it can also mean that the Workload Balancing service is not running or that there is a problem with the database configuration file.

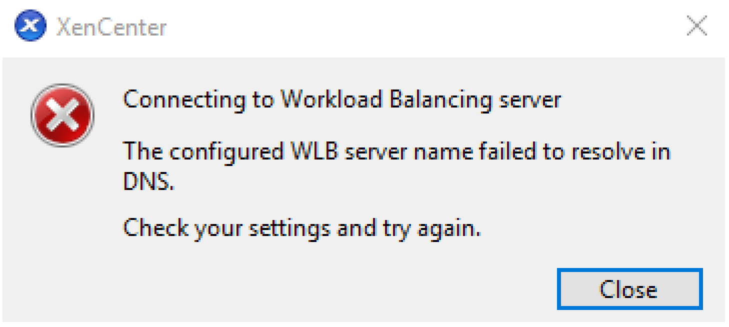
To fix credential issues, make sure that you are using the correct username and password. The default username for **WLB Server Credentials** field is `wlbuser` (not root). Root is the default administrator username. Note that `wlbuser` is not an actual user with logon privileges in the appliance (it does not exist under /`etc`/`passwd`) and thus these credentials are only used to connect to Workload

Balancing itself. As such, they can be easily reset by running the `wlbconfig` command. To change your credentials, see Change the Workload Balancing credentials. To run the `wlbconfig` command, you must be able to log into the appliance as root. If the root password is unknown, it can be reset using the regular CentOS/RHEL password recovery procedure.

If you have reset your credentials but the error still persists:

1. Check if the Workload Balancing process is running by using the `systemctl status workloadbalancing` command.
2. Make sure the `wlb.conf` file exists and is in the right directory by running this command: `cat /opt/vpx/wlb/wlb.conf`

Scenario 3



XenCenter

Connecting to Workload Balancing server

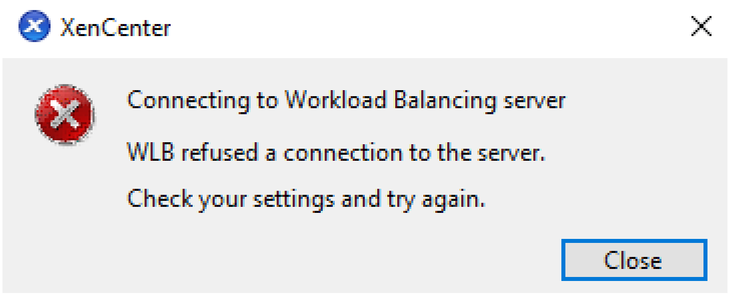
The configured WLB server name failed to resolve in DNS.

Check your settings and try again.

This indicates that there is an issue connecting to the port specified under the Server Address options when connecting to Workload Balancing from XenCenter (either the incorrect port was entered or the port is not listening). To troubleshoot this:

1. Make sure the target appliance is up and running.
2. Double-check the port entered on the Workload Balancing connection details window (default is 8012).
3. Make sure this port is enabled in the appliance and listening. Use commands like `telnet <port>` or `iptables -L` to help determine if the port is listening or if traffic is being denied on this port.

Scenario 4



XenCenter

Connecting to Workload Balancing server

WLB refused a connection to the server.

Check your settings and try again.

This error occurs when there is a problem with stunnel (either it's not running or the certificate/key pair is incorrect). To troubleshoot this, first verify the certificate and key:

1. Confirm the certificate has not expired by running the following command:

   ```
   openssl x509 -dates -in $(grep cert\ = /etc/stunnel/stunnel.conf |
       cut -d '=' -f2) -noout
   <!--NeedCopy-->
   ```

2. Compare the hex on the output of the following 2 commands. If the output does not match then the wrong key is being used.

   ```
   openssl x509 -modulus -in $(grep cert\ = /etc/stunnel/stunnel.conf
        |cut -d '=' -f2) -noout | openssl md5
   <!--NeedCopy-->
   ```

   and

   ```
    openssl rsa -modulus -in $(grep key\ = /etc/stunnel/stunnel.conf
       | cut -d '=' -f2) -noout | openssl md5
   <!--NeedCopy-->
   ```

If there are no problems with the certificate and key, make sure stunnel is running and is bound to port 8012 (or the configured port):

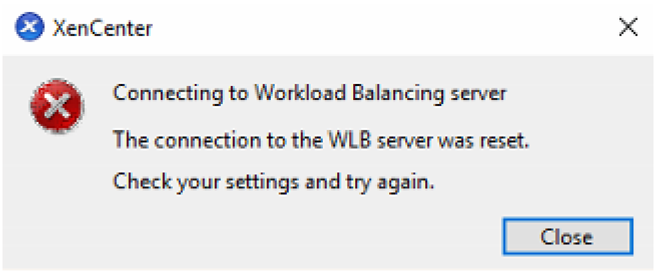
1. Run the following command in the WLB appliance CLI:

   ```
   netstat -tulpn
   <!--NeedCopy-->
   ```

   On the output, 8012 (or the custom port) should show `status: LISTEN`.

2. If the appliance ran out of space, stunnel won't run. Use commands like `df -h` or `du -hs /*` to see whether you have enough space available on your appliance. To increase the disk space, see Extend the virtual appliance disk.

Scenario 5



Connecting to Workload Balancing server

The connection to the WLB server was reset.

Check your settings and try again.

This error can occur because the stunnel process was terminated. If restarting the process yields the same results, restart the Workload Balancing virtual appliance.

Any other errors

If you encounter any other errors when attempting to connect to Workload Balancing or need further assistance performing the steps above, collect the Workload Balancing logs which can be found under the `/var/log/wlb` directory in the Workload Balancing appliance.

Contact Support for further assistance.

Workload Balancing stops working

If Workload Balancing doesn't work (for example, it doesn't let you save changes to settings), check the Workload Balancing log file for the following error message:

```
dwmdatacolsvc.exe: Don't have a valid pool. Trying again in 10 minutes.
<!--NeedCopy-->
```

This error typically occurs in pools that have one or more problematic VMs. When VMs are problematic, you might see the following behavior:

- **Windows**. The Windows VM crashes due to a stop error ("blue screen").
- **Linux**. The Linux VM might be unresponsive in the console and typically does not shut down.

To work around this issue:

1. Force the VM to shut down. To do so, you can do one of the following on the host with the problematic VM:

    - In XenCenter, select the VM, and then from the VM menu, click **Force Shutdown**.
    - Run the `vm-shutdown` xe command with the force parameter set to `true`. For example:

      ```
      xe vm-shutdown  force=true  uuid=vm_uuid
      <!--NeedCopy-->
      ```

      You can find the host UUID on the **General** tab for that host (in XenCenter) or by running the `host-list` xe command. You can find the VM UUID in the **General** tab for the VM or by running the `vm-list` xe command. For more information, see Command line interface.

2. In the `xsconsole` of the XenServer serving the crashed VM or in XenCenter, migrate all VMs to another host, then run the `xe-toolstack-restart` command. (Do not restart the toolstack while HA is enabled. If possible, temporarily disable HA before restarting the toolstack.)

Issues changing Workload Balancing servers

If you connect a pool to a different Workload Balancing server without disconnecting from Workload Balancing, both old and new Workload Balancing servers monitor the pool.

To solve this problem, you can take one of the following actions:

- Shut down and delete the old Workload Balancing virtual appliance.
- Manually stop the Workload Balancing services. These services are analysis, data collector, and Web service.

> **Note:**
>
> Do not use the `pool-deconfigure-wlb` xe command to disconnect a pool from the Workload Balancing virtual appliance or use the `pool-initialize-wlb` xe command to specify a different appliance.

XenServer Conversion Manager

March 6, 2024

Use the XenServer Conversion Manager virtual appliance to migrate your VMware ESXi/vCenter VMs to XenServer quickly and efficiently. You can convert up to 10 VMware ESXi/vCenter VMs in parallel at the same time. After converting your VMs, the Conversion Manager automatically shuts down by itself, saving resources on the host.

As part of the migration, XenCenter helps you prepare the VMs for networking and storage connectivity. After converting your VMs to a XenServer environment, they're almost ready to run.

Overview

XenServer allows you to:

- Convert up to 10 VMware ESXi/vCenter VMs in parallel using one simple wizard
- Map network settings between VMware and XenServer so your converted VMs can be up and running with the proper network settings
- Select a storage location where you would like your new XenServer VMs to run

> **Notes:**
>
> - XenCenter does not remove or change your existing VMware environment. VMs are duplicated onto your XenServer environment and not removed from VMware.
> - XenServer Conversion Manager virtual appliance supports converting VMware ESXi/vCenter VMs with different storage such as thin provisioning, thick provisioning, IDE, and SCSI.
> - XenServer Conversion Manager virtual appliance does not require the source VMs to have

> VMware Tools installed. You can perform conversion on VMware ESXi/vCenter VMs regardless of whether they have VMware Tools installed.
>
> - XenServer Conversion Manager virtual appliance cannot convert VMware ESXi/vCenter VMs with four or more disks into XenServer VMs. Your VMware ESXi/vCenter VMs must have three or fewer disks.

Understand XenServer

Before you can convert your environment, it is suggested that you become familiar with XenServer concepts. For more information, see Technical overview.

To successfully convert VMware ESXi/vCenter VMs to XenServer, perform the following tasks:

- Set up a basic XenServer environment, including installing XenServer. For more information, see Quick start and Install.
- Create a network in XenServer, assigning an IP address to a NIC. For more information, see Quick start.
- Connect to storage. For more information, see Quick start.

Compare VMware and XenServer terminology The following table lists the approximate XenServer equivalent for common VMware features, concepts, and components:

VMware Term	XenServer Equivalent
VMware vSphere Client	XenCenter (the management console for XenServer)
Cluster / Resource Pool	Resource Pool
Data Store	Storage Repository
vMotion	Live migration
Distributed Resource Scheduling (DRS)	Workload Balancing
High Availability (HA)	High Availability (HA)
vCenter Converter	XenServer Conversion Manager virtual appliance
Role Based Access Control (RBAC)	Role Based Access Control (RBAC)

Conversion overview

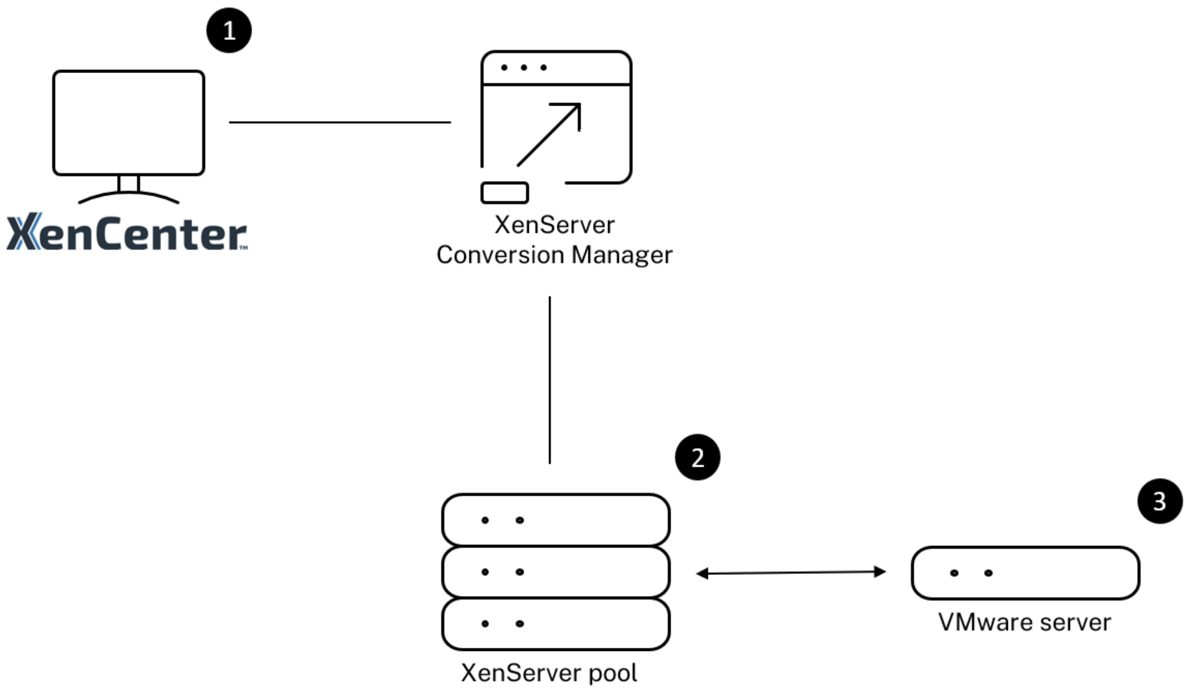
XenCenter and XenServer Conversion Manager virtual appliance create a copy of each targeted VM. After converting the targeted VM to a XenServer VM with comparable networking and storage connectivity, XenCenter imports the VM into your XenServer pool or host. You can convert as few as one or two VMs or perform batch conversions of an entire environment of up to 10 VMware ESXi/vCenter VMs in parallel at the same time.

> **Note:**
>
> Before converting the VMs from vSphere, you must shut down the VMs (intended for conversion) on vSphere. XenServer Conversion Manager virtual appliance does not support converting a running VM using memory copied from vSphere to XenServer.
>
> Also, before converting, ensure that a network and a storage controller exist in your VMware VM.

The conversion process requires four items:

- **XenCenter** - the XenServer management interface includes a conversion wizard where you set conversion options and control conversion. You can install XenCenter on your Windows desktop. XenCenter must be able to connect to XenServer and the XenServer Conversion Manager virtual appliance.
- **XenServer Conversion Manager virtual appliance** - a pre-packaged VM you import into the XenServer host or pool where you want to run the converted VMs. The virtual appliance converts the copies of the VMware ESXi/vCenter VMs into XenServer virtual machine format. After conversion, it imports these copies into the XenServer pool or host.
- **XenServer standalone host or pool** - the XenServer environment where you want to run the converted VMs.
- **VMware server** - XenServer Conversion Manager requires a connection to a VMware server that manages the VMs you want to convert. This connection can be to a vCenter Server, ESXi Server, or ESX Server. The VMs are not removed from the VMware server. Instead, the XenServer Conversion Manager Virtual Appliance makes a copy of these VMs and converts them to XenServer virtual-machine format.

The following illustration shows the relationships between these components:



> This illustration shows:
>
> 1. How XenCenter communicates with XenServer Conversion Manager virtual appliance.
> 2. How the XenServer Conversion Manager virtual appliance authenticates with the VMware server.
> 3. How the VMware server responds to the XenServer Conversion Manager virtual appliance during conversion.

The VMware server communicates with the XenServer Conversion Manager virtual appliance only when the appliance queries the VMware server for environment information and disk data throughout the conversion.

Summary of how to convert VMs You can configure the XenServer Conversion Manager virtual appliance and start to convert VMs in just a few easy steps:

1. Download the XenServer Conversion Manager virtual appliance from the XenServer downloads page.
2. Import the XenServer Conversion Manager virtual appliance into XenServer using XenCenter.
3. Configure the XenServer Conversion Manager virtual appliance by using XenCenter.
4. From XenCenter, launch the conversion wizard and start to convert VMs.
5. Complete the post-conversion tasks which include installing XenServer VM Tools for Windows on your Windows VMs. For Linux VMs, the XenServer Conversion Manager automatically installs

XenServer VM Tools for Linux during the conversion process.

After converting your VMs, the Conversion Manager automatically shuts down by itself, saving resources on the host. For more information on how to convert VMware ESXi/vCenter VMs, see Get started with Conversion Manager.

What's new in XenServer Conversion Manager

March 14, 2024

The latest version of the XenServer Conversion Manager virtual appliance is version 8.3.1. You can download this version of the XenServer Conversion Manager virtual appliance from the XenServer Downloads page.

If you already have a previous version of the XenServer Conversion Manager virtual appliance installed and wish to upgrade to the latest version, there is no automatic upgrade path. Download the latest version of the virtual appliance and remove the older version from your system.

What's new in 8.3.1

Released Feb 01, 2024

This update includes the following improvement:

- You can now convert up to 10 VMware ESXi/vCenter VMs in parallel at the same time.

Get started with XenServer Conversion Manager

April 18, 2024

You can easily convert your VMware ESXi/vCenter virtual machines (VMs) to XenServer in just a few steps:

1. Prepare your XenServer environment and review the prerequisite information.
2. Import and configure the XenServer Conversion Manager virtual appliance by using XenCenter.

   > **Note:**
   >
   > If you already have a previous version of the XenServer Conversion Manager virtual appliance installed and wish to upgrade to the latest version, there is no automatic upgrade

> path. Download the latest version of the virtual appliance from the XenServer downloads page and remove the older version from your system.

3. From XenCenter, launch the conversion wizard and begin converting your VMware ESXi/vCenter VMs to XenServer.
4. Complete the post-conversion tasks.
5. Review other conversion tasks.

Prepare your environment

Before converting your VMware environment, you must create and prepare the target XenServer standalone host or pool to run the converted VMware ESXi/vCenter VMs. Preparing your environment includes the following activities:

1. Defining a strategy of how you convert your VMware environment. Do you want to convert 1 or 2 VMs? Do you want to convert your entire environment? Do you want to create a pilot first to ensure that your configuration is correct? Do you run both environments in parallel? Do you want to maintain your existing cluster design when you convert to XenServer?
2. Planning your networking configuration. Do you want to connect to the same physical networks? Do you want to simplify or change your networking configuration?
3. Installing XenServer on the hosts you want in the pool. Ideally, plug the NICs on the hosts into their physical networks before you begin installation.
4. Creating a pool and performing any basic networking configuration. For example, do the following:
   - Configure a network to connect to the VMware cluster on the XenServer host (if the cluster is not on the same network as the XenServer host).
   - Configure a network to connect to the storage array. That is, if you use IP-based storage, create a XenServer network that connects to the physical network of the storage array.
   - Create a pool and add hosts to this pool.
5. (For shared storage and XenServer pools.) Preparing the shared storage where you store the virtual disks and creating a connection to the storage, known as a Storage Repository (SR) on the pool.
6. (Optional.) Although not a requirement for conversion, you might want to configure the administrator accounts on the XenServer pool to match those accounts on the VMware server. For information about configuring Role-based Access Control for Active Directory accounts, see Role-based access control.

Install XenServer and create a pool

Before you can convert VMware ESXi/vCenter VMs, ensure that you create a XenServer pool or host where you want to run the converted VMs. This pool must have networking configured so it can connect to the VMware server. You might also want to configure the same physical networks on the XenServer pool that you have in the VMware cluster, or simplify your networking configuration. If you want to run the converted VMs in a pool, create a storage repository before conversion and add the shared storage to the pool.

If you are new to XenServer, you can learn about XenServer basics, including basic installation and configuration, by reading Quick start.

XenServer environment considerations

Before installing XenServer and importing the virtual appliance, consider the following factors that might change your conversion strategy:

Selecting the host where you want to run the XenServer Conversion Manager virtual appliance. Import the virtual appliance into the stand-alone host or into a host in the pool where you run the converted VMs.

For pools, you can run the virtual appliance on any host in the pool, provided its storage meets the storage requirements.

> **Note:**
>
> We recommend that you run only one XenServer Conversion Manager in a pool at a time.

The storage configured for the pool or host where you want to run the converted VMs must meet specific requirements. If you want to run your newly converted VMs in a pool, their virtual disks must be stored on shared storage. However, if the converted VMs run on a single standalone host (not a pool), their virtual disks can use local storage.

If you want to run the converted VMs in a pool, ensure that you add the shared storage to the pool by creating a storage repository.

Guest operating systems supported for conversion:

You can convert VMware ESXi/vCenter VMs running the following Windows guest operating systems:

- Windows 10 (64-bit) Enterprise edition
- Windows Server 2016 (64-bit) Standard (Desktop) edition
- Windows Server 2019 (64-bit) Standard (Desktop) edition
- Windows Server 2022 (64-bit) Standard (Desktop) edition

> **Note:**
>
> Only the listed Windows SKUs are supported for conversion.

The following Linux operating systems are also supported:

- Red Hat Enterprise Linux 7.9 (64-bit) with the following configuration:
  - File system: EXT3 or EXT4
  - Boot partition type: btrfs, lvm, or plain
- Red Hat Enterprise Linux 8.x (64-bit) with the following configuration:
  - File system: EXT3 or EXT4
  - Boot partition type: lvm or plain
- Ubuntu 20.04 with the following configuration:
  - File system: EXT3 or EXT4
  - Boot partition type: lvm or regular

For more information about the guest operating systems supported by XenServer, see Guest operating system support.

Meet networking requirements To convert VMware ESXi/vCenter VMs, the XenServer Conversion Manager virtual appliance needs connectivity to a physical network or VLAN that can contact the VMware server. (In the following sections, this network is referred to as the "VMware network".)

If the VMware server is on a different physical network than the hosts in the XenServer pool, add the network to XenServer before conversion.

> **Note:**
>
> - The time it takes for your VMs to be converted depends on the physical distance between your VMware and XenServer networks and also the size of your VM's virtual disk. You can estimate how long the conversion will last by testing the network throughput between your VMware server and XenServer.
> - By default, the XenServer Conversion Manager uses HTTPS to download the VM's virtual disk during VM conversion. To speed up the migration process, you can switch the download path to HTTP. For more information, see VMware's article Improving transfer speed of task with library items.

Map your existing network configuration XenServer Conversion Manager virtual appliance includes features that can reduce the amount of manual networking configuration needed after you convert from your existing VMware ESXi/vCenter VMs to XenServer. For example, XenServer Conversion Manager virtual appliance will:

- Preserve virtual MAC addresses on the VMware ESXi/vCenter VMs and reuse them in the resulting XenServer VMs. Preserving the MAC addresses associated with virtual network adapters (virtual MAC addresses) may:
  - Help preserve IP addresses in environments using DHCP
  - Be useful for software programs whose licensing references the virtual MAC addresses
- Map (virtual) network adapters. XenServer Conversion Manager virtual appliance can map VMware networks onto XenServer networks so that after the VMs are converted, their virtual network interfaces are connected accordingly.

  For example, if you map VMware 'Virtual Network 4'to XenServer 'Network 0', any VMware VM that had a virtual adapter connected to 'Virtual Network 4'is connected to 'Network 0'after conversion. XenServer Conversion Manager virtual appliance does not convert or migrate any hypervisor network settings. The wizard only alters a converted VM's virtual network interface connections based on the mappings provided.

  > **Note:**
  >
  > You do not need to map all of your VMware networks on to the corresponding XenServer networks. However, if you prefer, you can change the networks the VMs use, reduce, or consolidate the number of networks in your new XenServer configuration.

  To gain the maximum benefit from these features, we recommend the following:

  - Before installing XenServer, plug the hosts into the networks on the switch (that is, the ports) that you would like to configure on the host.
  - Ensure that the XenServer pool can see the networks that you would like to be detected. Specifically, plug the XenServer hosts into switch ports that can access the same networks as the VMware cluster.

  Though it is easier to plug the XenServer NICs into the same networks as the NICs on the VMware hosts, it is not required. If you would like to change the NIC/network association, you can plug a XenServer NIC into a different physical network.

Prepare for the XenServer Conversion Manager virtual appliance networking requirements
When you perform a conversion, you must create a network connection to the network where the VMware server resides. XenServer Conversion Manager virtual appliance uses this connection for conversion traffic between the XenServer host and the VMware server.

To create this network connection, you must perform two tasks:

- When you import the XenServer Conversion Manager virtual appliance, specify the network you added for conversion traffic as a virtual network interface. You can do so by configuring **interface 1** so it connects to that network.
- Before you run the conversion wizard, add the network connecting VMware and XenServer to the XenServer host where you want to run the converted VMs.

By default, when you import the XenServer Conversion Manager virtual appliance, XenCenter creates one virtual network interface associated with Network 0 and NIC0 (eth0). When adding a network for conversion, select a network other than XenServer's management network to improve performance in busy pools. For more information about the management interface, see Networking.

Inside the XenServer conversion manager, you might see multiple network interfaces (eth0 and eth1). eth0 attaches to the host's internal network which is used to communicate with the local dom0. eth1 attaches to the routable network which is used to communicate with XenCenter.

To add a network to XenServer:

1. In the **Resource** pane in XenCenter, select the pool where you would like to run XenServer Conversion Manager virtual appliance.
2. Click the **Networking** tab.
3. Click **Add Network**.
4. On the **Select Type** page, select **External Network**, and click **Next**.
5. On the **Name** page, enter a meaningful name for the network (for example, "VMware network") and a description.
6. On the **Interface** page, specify the following:
   - **NIC**. The NIC that you want XenServer to use to create the network. Select the NIC that is plugged in to the physical or logical network of the VMware server.
   - **VLAN**. If the VMware network is a VLAN, enter the VLAN ID (or "tag").
   - **MTU**. If the VMware network uses jumbo frames, enter a value for the Maximum Transmission Unit (MTU) between 1500 and 9216. Otherwise, leave the MTU box its default value of 1500.

     > **Note:**
     >
     > Do not select the **Automatically add this network to new virtual machines** check box.
7. Click **Finish**.

Meet storage requirements Before you convert batches of VMware ESXi/vCenter VMs, consider your storage requirements. Converted VM disks are stored on a XenServer storage repository.

This storage repository must be large enough to contain the virtual disks for all the converted VMs you want to run in that pool. For converted machines that only run on a standalone host, you can specify either local or shared storage as the location for the converted virtual disks. For converted machines running in pools, you can only specify shared storage.

To create a storage repository:

1. In the **Resource** pane in XenCenter, select the pool where you intend to run the XenServer Conversion Manager virtual appliance.
2. Click the **Storage** tab.
3. Click **New SR** and follow the instructions in the wizard. For more instructions, press **F1** to display the online help.

XenServer requirements You can run VMs converted with this release of XenServer Conversion Manager on the following versions of XenServer:

- XenServer 8
- Citrix Hypervisor 8.2 Cumulative Update 1

VMware requirements XenServer Conversion Manager virtual appliance can convert VMware ESXi/vCenter VMs from the following versions of VMware:

- vCenter Server 6.7.x, 7.x, and 8.x
- vSphere 6.7.x, 7.x, and 8.x
- ESXi 6.7.x, 7.x, and 8.x

> **Note:**
>
> XenServer Conversion Manager virtual appliance cannot convert VMware ESXi/vCenter VMs with four or more disks into XenServer VMs. Your VMware ESXi/vCenter VMs must have three or fewer disks.
>
> Your VMware ESXi/vCenter VMs must also have a network and a storage controller configured.

Prepare to import the virtual appliance Before importing the virtual appliance, note the following information and make the appropriate changes to your environment, as applicable.

Download the virtual appliance The XenServer Conversion Manager virtual appliance is packaged in XVA format. You can download the virtual appliance from the XenServer downloads page. When downloading the file, save it to a folder on your local hard drive (typically, but not necessarily, on the computer where XenCenter is installed). After the `.xva` file is on your hard drive, you can import it into XenCenter.

Virtual appliance prerequisites The XenServer Conversion Manager virtual appliance requires a minimum of:

- Citrix Hypervisor 8.2 Cumulative Update 1, XenServer 8
- Disk space: 30 GB of disk space
- Memory: 6 GB
- Virtual CPU allocation: 2 vCPU

Import and configure the virtual appliance

The XenServer Conversion Manager virtual appliance is a single pre-installed VM designed to run on a XenServer host. Before importing it, review the prerequisite information and considerations in the section called *Preparing to import the virtual appliance*.

Import the virtual appliance into XenServer

To import the XenServer Conversion Manager virtual appliance into the pool or host where you want to run the converted VMs, use the XenCenter **Import** wizard:

1. Open XenCenter. Right-click on the pool (or host) into which you want to import the virtual appliance package, and select **Import**.
2. Browse to locate the virtual appliance package.
3. Select the pool or a *home server* where you want to run the XenServer Conversion Manager virtual appliance.

   > **Note:**
   >
   > A home server is the host that provides the resources for a VM in a pool. While it can, a XenServer attempts to start the VM on that host, before trying other hosts. If you select a host, the XenServer Conversion Manager virtual appliance uses this host as its home server. If you select the pool, the virtual appliance automatically starts on the most suitable host in that pool.

4. Choose a storage repository on which to store the virtual disk for the XenServer Conversion Manager virtual appliance and then click **Import**. To add a storage repository to the pool, see the section called "Meet Storage Requirements."You can choose either local or shared storage.
5. Ensure the network to be used for conversion (which connects the VMware server to the XenServer host) is selected as the network associated with **interface 1** ("virtual NIC 1").
    - If the correct network does not appear beside interface 1, use the list in the **Network** column to select a different network.
    - If you have not added the VMware network that is on a different physical network than the pool, do the following:
        a) Exit the wizard.
        b) Add the network to the pool.
        c) Rerun the wizard.

      For more information, see **To add a network to XenServer**.

      > **Warning:**
      >
      > Do NOT configure NIC0 to your customer network. Assign NIC0 only to "Host internal management network."
6. Leave the **Start VM after import** check box enabled, and click **Finish** to import the virtual appliance.
7. After importing the `.xva` file, the XenServer Conversion Manager virtual appliance appears in the **Resources** pane in XenCenter.

Configure the XenServer Conversion Manager virtual appliance

Before you can use the XenServer Conversion Manager virtual appliance to convert VMware ESXi/vCenter VMs, configure it using the XenCenter **Console** tab:

1. After importing the XenServer Conversion Manager virtual appliance, click the **Console** tab.
2. Read the license agreement. To view the contents of the license agreement, open the URL in a web browser. Press any key to continue.
3. Enter and confirm a new root password for the XenServer Conversion Manager virtual appliance. We recommend selecting a strong password.
4. Enter a host name for the XenServer Conversion Manager virtual appliance.
5. Enter the domain suffix for the virtual appliance. For example, if the fully qualified domain name (FQDN) for the virtual appliance is `citrix-migrate-vm.domain4.example.com`, enter `domain4.example.com`.

6. Enter **y** to use DHCP to obtain the IP address automatically for the XenServer Conversion Manager virtual appliance. Otherwise, enter **n** and then enter a static IP address, subnet mask, and gateway for the VM.
7. Review the host name and network setting and enter **y** when prompted. This step completes the XenServer Conversion Manager virtual appliance configuration process.
8. When you have successfully configured the appliance, a login prompt appears. Enter the login credentials and press Enter to log in to the XenServer Conversion Manager virtual appliance.

Convert VMware ESXi/vCenter VMs

When you convert VMware ESXi/vCenter VMs, they are imported into the XenServer pool or standalone host where you are running the XenServer Conversion Manager virtual appliance. Converted VMs retain their original VMware settings for the virtual processor and virtual memory.

Before you start the conversion procedure, ensure that the following is true:

- You have the credentials for the XenServer pool (or standalone host). Either the root account credentials or a Role-Based Access Control (RBAC) account with the Pool Admin role configured is acceptable.
- You have the credentials for the VMware server containing the VMs you want to convert. The conversion procedure requires you connect the XenServer Conversion Manager Console to the VMware server.
- The VMware virtual machines to convert are powered off.
- The VMware virtual machines to convert have a network and a storage controller configured.
- The XenServer pool (or host) that run the converted VMs is connected to a storage repository. The storage repository must contain enough space for the converted virtual disks.
- If you want to run your newly converted VMs in a pool, the storage repository must be shared storage. However, if the converted VMs run on a single standalone host (not a pool), you can use local storage.
- The virtual disks of the VM to convert are less than 2 TiB.
- XenServer pool (or host) has networks that the converted VMs use.

To convert your VMware ESXi/vCenter VMs into VMs that can run in a XenServer environment:

1. Ensure that the virtual appliance is installed and running on the XenServer host or pool where you want to import the VMs.
2. In XenCenter, go to **Pool** > **Conversion Manager**.

   The **Conversion Manager** window opens. Wait while the wizard connects to your virtual appliance.
3. Click **New Conversion**.

4. In the **New Conversion** wizard, enter the credentials for the VMware server:

    - **Server**. Enter the IP address or FQDN for the VMware server that contains the VMs you want to convert to XenServer.
    - **Username**. Enter a valid user name for this VMware server. This account must either be a VMware admin account or have a Root role.
    - **Password**. Enter the password for the user account you specified in the **Username** box.

    Click **Next**. XenCenter connects to the VMware server.

5. In the **Virtual Machines** page, select from the list of VMs hosted in the VMware server the VMs that you want to convert. Click **Next**.

6. In the **Storage** page, select the storage repository you want to use during conversion. This storage repository is where the VMs and the virtual disks that you are creating are stored permanently.

    This tab indicates the proportion of available storage that the virtual disks of the converted VMs consume.

7. On the **Networking** page, for each VMware network listed, select the XenServer network to map it to. You can also select whether to preserver virtual MAC addresses. Click **Next**.

8. Review the options you configured for the conversion process. You can click **Previous** to change these options. To proceed with the configuration shown, click **Finish**.

    The conversion process begins. Conversion from ESXi or vSphere can take several minutes depending on the size of the virtual disks.

    After converting your VMs, the Conversion Manager automatically shuts down by itself, saving resources on the host. Start a VM by selecting the VM's host and then clicking **Pool** > **Conversion Manager**.

The **Conversion Manager** window displays conversions in progress and completed conversions.

Steps after conversion

For Windows VMs, you must install XenServer VM Tools for Windows. For Linux VMs, you do not need to install XenServer VM Tools for Linux as the Conversion Manager automatically installs it during the conversion process.

After conversion, in XenCenter perform the following steps on your newly converted VMs:

On Windows Machines:

1. On Windows VMs, depending on your Microsoft licensing model, you might have to reactivate the VM's Windows license. This reactivation happens because the Windows operating system perceives the conversion as a hardware change.

2. On Windows VMs, install XenServer VM Tools for Windows to obtain high-speed I/O for enhanced disk and network performance. XenServer VM Tools for Windows also enable certain functions and features, including cleanly shutting down, rebooting, suspending, and live migrating VMs. You can download the XenServer VM Tools for Windows from the XenServer downloads page.

If you are working with a VM that does not have XenServer VM Tools installed, a XenServer VM Tools not installed message appears on the **General** tab in the **General** pane.

> **Note:**
>
> XenServer VM Tools for Windows must be installed on each Windows VM for the VM to have a fully supported configuration. Although Windows VMs function without XenServer VM Tools for Windows, their performance can be impacted.

Enable VNC On Linux machines

On Linux VMs, configure the VNC server. For more information, see Enable VNC for Linux VMs.

> **Note:**
>
> The VNC password must have at least six characters.

Other conversion tasks

The **Manage Conversions** window enables you to perform other tasks related to converting VMs. These tasks include clearing jobs, saving a summary of jobs, retrying jobs, canceling jobs, and displaying the log file.

To clear all jobs:

1. Select **Clear All**.
2. When prompted to confirm this action, click **Yes** to continue.

To save a summary of jobs:

1. Click **Export All**.
2. Specify where to save the CSV file.
3. Click **Save**.

To retry a job:

1. Select the job from the list.
2. Click **Retry**.

> **Note:**
>
> The **Retry** option is only enabled for failed or canceled jobs.

To cancel a job:

1. Select the job from the list.
2. Click **Cancel**.

> **Note:**
>
> Cancel jobs is only enabled for queued or running jobs.

To save the conversion log file for a single job:

1. Select the job from the list.
2. From the logs menu, Click **Fetch Selected Log**.
3. Specify where to save the log file.

To save the conversion log file for all jobs:

1. From the logs menu, Click **Fetch All Logs**.
2. Specify where to save the log file.

To display conversion details:

1. Select the job from the list.

   The information is displayed in the **Details** panel.

Troubleshoot XenServer Conversion Manager

March 6, 2024

This section provides information about troubleshooting the conversion process and converted VMs.

Problems starting a converted VM

In general, conversion runs smoothly and XenServer Conversion Manager virtual appliance converts VMs without any issues. However, in some rare cases, you might receive errors when attempting to open converted VMs. The following sections provide some guidance on resolving errors and other issues.

Blue screen with Windows STOP code 0x0000007B

This stop code indicates that XenServer Conversion Manager virtual appliance was unable to configure a Windows device that is critical to boot in XenServer for the first time. Save the logs and send them to Support for further guidance.

Windows product activation

Depending on your licensing model, an error message on system activation might appear when you attempt to start a Windows VM.

Lost network settings in a Windows VM

If you import a Windows VM from an ESXi server to XenServer, the IPv4/IPv6 network settings can be lost. To retain the network settings, reconfigure the IPv4/IPv6 settings after completing the conversion.

Unable to start VMware SCSI disk

If a VMware VM boots from a SCSI disk but also has IDE hard disks configured, the VM might not boot when you convert it to XenServer. This issue occurs because the migration process assigns the IDE hard disks lower device numbers than SCSI disks. However, XenServer boots from the hard disk assigned to device 0. To resolve this issue, rearrange the virtual disk position in XenCenter so that the VM reboots from the virtual disk that contains the operating system.

To change the position of the virtual disk containing the operating system:

1. In the XenCenter **Resources** pane, select the powered off guest VM.
2. Select the **Storage** tab.
3. From the **Virtual Disks** list, select the virtual disk containing the operating system and then click **Properties**.
4. In the virtual disk's **Properties** dialog, click the ***vm_name*** tab to display device options.
5. From the **Device Position** list, select **0** and Click **OK**.

Problems during conversion

If you experience problems or errors when converting VMs, try exporting the VMware VM as an OVF package. If you cannot export the VMware VM as an OVF package, Conversion Manager cannot convert this VM. Use the error messages you receive when attempting to export the VM as an OVF package to troubleshoot and fix the issues with your VMware VM. For example, you might have to configure a network or a storage controller before the VM can be exported as an OVF package or converted. For more information about troubleshooting your VMware ESXi/vCenter VMs, see the VMware documentation.

If you see any errors when converting Linux VMs, remove the converted VM, restart the XenServer Conversion Manager virtual appliance and retry.

Logs of failed conversions are stored in the XenServer Conversion Manager virtual appliance and can be retrieved by clicking **Fetch All Logs** on the **Conversion Manager** window. When you contact Support to raise any issues, we recommend that you provide the conversion log file and, additionally, a full server status report for troubleshooting. For more information, see Creating a Server Status Report.

Command-line interface

April 17, 2024

The *xe* CLI enables you to script and automate system administration tasks. Use the CLI to integrate XenServer into an existing IT infrastructure.

Getting started with the xe CLI

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

On your XenServer host

The xe command line interface is installed by default on your host. You can run xe CLI commands in the dom0 console. Access the dom0 console in one of the following ways:

- In XenCenter, go to the **Console** tab for the host where you want to run the command.
- SSH into the host where you want to run the command.

On Windows

On Windows, the `xe.exe` command is installed along with XenCenter.

To use the `xe.exe` command, open a Windows Command Prompt and change directories to the directory where the `xe.exe` file is located (typically `C:\Program Files (x86)\XenServer\XenCenter`). If you add the `xe.exe` installation location to your system path, you can use the command without having to change into the directory.

On Linux

On RPM-based distributions (such as Red Hat), you can install the stand-alone xe command from the RPM named `client_install/xapi-xe-BUILD.x86_64.rpm` on the main XenServer installation ISO.

To install from the RPM, use the following command:

```
rpm -ivh xapi-xe-BUILD.x86_64.rpm
<!--NeedCopy-->
```

You can use parameters at the command line to define the XenServer host, user name, and password to use when running xe commands. However, you also have the option to set this information as an environment variable. For example:

```
export XE_EXTRA_ARGS="server=<host name>,username=<user name>,password
    =<password>"
<!--NeedCopy-->
```

> **Note:**
>
> The remote xe CLI on Linux might hang when attempting to run commands over a secure connection and these commands involve file transfer. If so, you can use the `--no-ssl` parameter to run the command over an insecure connection to the XenServer host.

Getting help with xe commands

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

```
xe help command
<!--NeedCopy-->
```

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

```
xe help
<!--NeedCopy-->
```

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

```
xe help --all
<!--NeedCopy-->
```

Basic xe syntax

The basic syntax of all XenServer xe CLI commands is:

```
xe command-name argument=value argument=value
<!--NeedCopy-->
```

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 assumes default values for some of the optional arguments when invoked without them.

If the xe command runs remotely, extra arguments are used to connect and authenticate. These arguments also take the form `argument=argument_value`.

The `server` argument is used to specify the host name 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, the xe command attempts to read the password from the specified file and uses that password to connect. (Any trailing CRs and LFs at the end of the file are stripped off.) This method 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
<!--NeedCopy-->
```

Example: On a remote XenServer host:

```
xe vm-list username=username password=password server=hostname
<!--NeedCopy-->
```

Shorthand syntax is also available for remote connection arguments:

- `-u` user name
- `-pw` password
- `-pwf` password file
- `-p` port
- `-s` server

Example: On a remote XenServer host:

```
xe vm-list -u myuser -pw mypassword -s hostname
<!--NeedCopy-->
```

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

```
export XE_EXTRA_ARGS="server=jeffbeck,port=443,username=root,password=
    pass"
<!--NeedCopy-->
```

After running this command, you no longer have to specify the remote XenServer host parameters in each xe command that you run.

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. Do not include whitespace between the argument name, the equals sign (=), and the value. Any argument not conforming to this format is ignored.

For values containing spaces, write: `argument="value with spaces"`

When you use the CLI on your XenServer host, commands have a tab completion feature similar to the feature in the standard Linux bash shell. For example, if you type `xe vm-l` and then press the **TAB** key, the rest of the command is displayed. If more than one command begins with `vm-l`, pressing **TAB** a second time lists the possibilities. This feature is useful when specifying object UUIDs in commands.

> **Note:**
>
> Tab completion does not normally work when running commands on a remote XenServer host. However, if you set the XE_EXTRA_ARGS variable on the machine where you enter the commands, tab completion is enabled. For more information, see Basic xe syntax.

Command types

The CLI commands can be split in two halves. Low-level commands are concerned with listing and parameter manipulation of API objects. Higher level commands are used to interact 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`

Not every value of *class* has the full set of *class*-param-*action* commands. Some values of *class* have a smaller set of commands.

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`, a value in parentheses indicates whether parameters are read-write (RW) or read-only (RO).
The output of `xe vm-param-list` on a specified VM might have the following 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 piece of 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 parameter 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.

To filter on a map parameter or set a map parameter, use a colon (:) to separate 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
<!--NeedCopy-->
```

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 commands takes a `uuid` parameter to specify the particular object. Since these commands are considered low-level commands, they must use the UUID and not the VM name label.

- `xe class-param-list uuid=uuid`

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

- `xe class-param-get uuid=uuid param-name=parameter param-key=key`

  Returns the value of a particular parameter. For a map parameter, specifying the param-key gets the value associated with that key in the map. If param-key is not specified or if the parameter is a set, the command returns a string representation of the set or map.

- `xe class-param-set uuid=uuid param=value`

  Sets the value of one or more parameters.

- `xe class-param-add uuid=uuid param-name=parameter key=value param-key=key`

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

- `xe 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.

- `xe 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, this type of command lists all objects, printing a subset of the parameters. This behavior can be modified in the following ways:

- It can filter the objects so that it only outputs a subset
- The parameters that are printed can be modified.

To change the parameters that are printed, specify the argument *params* as a comma-separated list of the required parameters. For example:

```
xe vm-list params=name-label,other-config
<!--NeedCopy-->
```

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

```
xe vm-list params=all
<!--NeedCopy-->
```

The list command doesn't show some parameters that are expensive to calculate. These parameters are shown as, for example:

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

To obtain these fields, use either the command *class*-param-list or *class*-param-get

To filter the list, the CLI matches parameter values with those values 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
<!--NeedCopy-->
```

This command lists only those VMs for which *both* the field `power-state` has the value *halted* and the field `HVM-boot-policy` has the value *BIOS order*.

You can also filter the list by the value of keys in maps or by the existence of values in a set. The syntax for filtering based on keys in maps is `map-name:key=value`. The syntax for filtering based on values existing in a set is `set-name:contains=value`.

When 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 host with three VMs installed gives the three UUIDs of the VMs:

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

Secrets

XenServer provides a secrets mechanism to avoid passwords being stored in plaintext in command-line history or on API objects. XenCenter uses this feature automatically and it can also be used from the xe CLI for any command that requires a password.

> **Note:**
>
> Password secrets cannot be used to authenticate with a XenServer host from a remote instance of the xe CLI.

To create a secret object, run the following command on your XenServer host.

```
xe secret-create value=my-password
<!--NeedCopy-->
```

A secret is created and stored on the XenServer host. The command outputs the UUID of the secret object. For example, `99945d96-5890-de2a-3899-8c04ef2521db`. Append `_secret` to the name of the password argument to pass this UUID to any command that requires a password.

Example: On the XenServer host where you created the secret, you can run the following command:

```
    xe sr-create device-config:location=sr_address device-config:type=
       cifs device-config:username=cifs_username  \
    device-config:cifspassword_secret=secret_uuid name-label="CIFS ISO
       SR" type="iso" content-type="iso" shared="true"
<!--NeedCopy-->
```

Command history

Some xe commands, for example `xe vm-migrate` or `xe pool-enable-external-auth`, take secrets like passwords as parameters. These can end up in the shell history and during execution of the command are visible in the process table. It is therefore important to run these commands only in trustworthy environments.

For the bash shell, you can use the `HISTCONTROL` variable to control which commands are stored in the shell history.

xe command reference

This section groups the commands by the objects that the command addresses. These objects are listed alphabetically.

Appliance commands

Commands for creating and modifying VM appliances (also known as vApps). For more information, see vApps.

Appliance parameters

Appliance commands have the following parameters:

Parameter Name	Description	Type
`uuid`	The appliance uuid	Required
`name-description`	The appliance description	Optional
`paused`		Optional
`force`	Force shutdown	Optional

`appliance-assert-can-be-recovered`

```
xe appliance-assert-can-be-recovered uuid=appliance-uuid database:vdi-
    uuid=vdi-uuid
<!--NeedCopy-->
```

Tests whether storage is available to recover this VM appliance/vApp.

`appliance-create`

```
xe appliance-create name-label=name-label [name-description=name-
    description]
<!--NeedCopy-->
```

Creates an appliance/vApp. For example:

```
xe appliance-create name-label=my_appliance
<!--NeedCopy-->
```

Add VMs to the appliance:

```
xe vm-param-set uuid=VM-UUID appliance=appliance-uuid
<!--NeedCopy-->
```

`appliance-destroy`

```
xe appliance-destroy uuid=appliance-uuid
<!--NeedCopy-->
```

Destroys an appliance/vApp. For example:

```
xe appliance-destroy uuid=appliance-uuid
<!--NeedCopy-->
```

`appliance-recover`

```
xe appliance-recover uuid=appliance-uuid database:vdi-uuid=vdi-uuid [
    paused=true|false]
<!--NeedCopy-->
```

Recover a VM appliance/vApp from the database contained in the supplied VDI.

`appliance-shutdown`

```
xe appliance-shutdown uuid=appliance-uuid [force=true|false]
<!--NeedCopy-->
```

Shuts down all VMs in an appliance/vApp. For example:

```
xe appliance-shutdown uuid=appliance-uuid
<!--NeedCopy-->
```

`appliance-start`

```
xe appliance-start uuid=appliance-uuid [paused=true|false]
<!--NeedCopy-->
```

Starts an appliance/vApp. For example:

```
xe appliance-start uuid=appliance-uuid
<!--NeedCopy-->
```

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 *file name*	Required
`since`	Specific date/time point	Optional

`audit-log-get`

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

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
<!--NeedCopy-->
```

Bonding commands

Commands for working with network bonds, for resilience with physical interface failover. For more information, see Networking.

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 two or more physical interfaces that 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 main bond PIF	Read only
`members`	Set of UUIDs for the underlying bonded PIFs	Read only

`bond-create`

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

Create a bonded network interface on the network specified from a list of existing PIF objects. The command fails in any of the following cases:

- 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
- If fewer than 2 PIFs are supplied

bond-destroy

```
xe bond-destroy uuid=bond_uuid
<!--NeedCopy-->
```

Deletes a bonded interface specified by its UUID from a host.

bond-set-mode

```
xe bond-set-mode uuid=bond_uuid mode=bond_mode
<!--NeedCopy-->
```

Change the bond mode.

CD commands

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

CD parameters

CDs have the following parameters:

Parameter Name	Description	Type
`uuid`	Unique identifier/object reference for the CD	Read only
`name-label`	Name for the CD	Read/write
`name-description`	Description text for the CD	Read/write
`allowed-operations`	A list of the operations that can be performed on this CD	Read only set parameter
`current-operations`	A list of the operations that are currently in progress on this CD	Read only set parameter
`sr-uuid`	The unique identifier/object reference for the SR this CD is part of	Read only
`sr-name-label`	The name for the SR this CD is part of	Read only
`vbd-uuids`	A list of the unique identifiers for the VBDs on VMs that connect to this CD	Read only set parameter

Parameter Name	Description	Type
`crashdump-uuids`	Not used on CDs. Because crashdumps cannot be written to CDs	Read only set parameter
`virtual-size`	Size of the CD as it appears to VMs (in bytes)	Read only
`physical-utilisation`	Amount of physical space that the CD image takes 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`	Value is **`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`	Value is **`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 extra configuration parameters for the CD	Read/write map parameter
`location`	The path on which the device is mounted	Read only
`managed`	Value is **`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`	Value is **`true`** if this template is a CD snapshot	Read only
`snapshot_of`	The UUID of the CD that this template is a snapshot of	Read only

Parameter Name	Description	Type
`snapshots`	The UUIDs of any snapshots that have been taken of this CD	Read only
`snapshot_time`	The timestamp of the snapshot operation	Read only

`cd-list`

```
xe cd-list [params=param1,param2,...] [parameter=parameter_value]
<!--NeedCopy-->
```

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. When `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.

Cluster commands

Commands for working with clustered pools.

Clustered pools are resource pools that have the clustering feature enabled. Use these pools with GFS2 SRs. For more information, see Clustered pools

The cluster and cluster-host objects can be listed with the standard object listing commands (`xe cluster-list` and `xe cluster-host-list`), and the parameters manipulated with the standard parameter commands. For more information, see Low-level parameter commands.
Commands for working with clustered pools.

Cluster parameters

Clusters have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the cluster	Read only

Parameter Name	Description	Type
`cluster-hosts`	A list of unique identifiers/object references for the hosts in the cluster	Read only set parameter
`cluster-token`	The secret key used by `xapi-clusterd` when it talks to itself on other hosts	Read only
`cluster-stack`	The technology stack providing the clustering capabilities. Possible values are `corosync`.	Read only
`allowed-operations`	Lists the operations allowed in this state. This list is advisory only and the cluster state may have changed by the time a client reads this field.	Read only set parameter
`current-operations`	Lists the operations currently in process. This list is advisory only and the cluster state may have changed by the time a client reads this field.	Read only set parameter
`token-timeout`	The `corosync` token timeout in seconds	Read only
`token-timeout-coefficient`	The `corosync` token timeout coefficient in seconds	Read only
`pool-auto-join`	True if automatically joining new pool members to the cluster. This is set to **`true`**.	Read only
`cluster-config`	A list of key/value pairs that specify extra configuration parameters for the cluster.	Read only map parameter
`other-config`	A list of key/value pairs that specify extra configuration parameters for the cluster.	Read/write map parameter

`cluster-host-destroy`

```
xe cluster-host-destroy uuid=host_uuid
<!--NeedCopy-->
```

Destroy a cluster host, effectively leaving the cluster.

cluster-host-disable

```
xe cluster-host-disable uuid=cluster_uuid
<!--NeedCopy-->
```

Disable cluster membership for an enabled cluster host.

cluster-host-enable

```
xe cluster-host-enable uuid=cluster_uuid
<!--NeedCopy-->
```

Enable cluster membership for a disabled cluster host.

cluster-host-force-destroy

```
xe cluster-host-force-destroy uuid=cluster_host
<!--NeedCopy-->
```

Destroy a cluster host object forcefully, effectively leaving the cluster.

cluster-pool-create

```
xe cluster-pool-create network-uuid=network_uuid [cluster-stack=
    cluster_stack] [token-timeout=token_timeout] [token-timeout-
    coefficient=token_timeout_coefficient]
<!--NeedCopy-->
```

Create pool-wide cluster.

cluster-pool-destroy

```
xe cluster-pool-destroy cluster-uuid=cluster_uuid
<!--NeedCopy-->
```

Destroy pool-wide cluster. The pool continues to exist, but it is no longer clustered and can no longer use GFS2 SRs.

cluster-pool-force-destroy

```
xe cluster-pool-force-destroy cluster-uuid=cluster_uuid
<!--NeedCopy-->
```

Force destroy pool-wide cluster.

cluster-pool-resync

```
xe cluster-pool-resync cluster-uuid=cluster_uuid
<!--NeedCopy-->
```

Resync a cluster across a pool.

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. For more information, see Low-level parameter commands.

Console parameters

Consoles have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the console	Read only
`vm-uuid`	The unique identifier/object reference of the VM this console is open on	Read only
`vm-name-label`	The name of the VM this console is open on	Read only
`protocol`	Protocol this console uses. Possible values are `vt100`: VT100 terminal, `rfb`: Remote Framebuffer Protocol (as used in VNC), or `rdp`: Remote Desktop Protocol	Read only

Parameter Name	Description	Type
`location`	URI for the console service	Read only
`other-config`	A list of key/value pairs that specify extra configuration parameters for the console.	Read/write map parameter

`console`

```
xe console
<!--NeedCopy-->
```

Attach to a particular console.

Diagnostic commands

Commands for gathering diagnostic information from XenServer.

`diagnostic-compact`

```
xe diagnostic-compact
<!--NeedCopy-->
```

Perform a major GC collection and heap compaction.

`diagnostic-db-stats`

```
xe diagnostic-db-stats
<!--NeedCopy-->
```

Print database statistics.

`diagnostic-gc-stats`

```
xe diagnostic-gc-stats
<!--NeedCopy-->
```

Print GC statistics.

diagnostic-license-status

```
xe diagnostic-license-status
<!--NeedCopy-->
```

Help diagnose pool-wide licensing problems.

diagnostic-net-stats

```
xe diagnostic-net-stats [uri=uri] [method=method] [params=param1,param2
    ...]
<!--NeedCopy-->
```

Print network statistics.

diagnostic-timing-stats

```
xe diagnostic-timing-stats
<!--NeedCopy-->
```

Print timing statistics.

diagnostic-vdi-status

```
xe diagnostic-vdi-status uuid=vdi_uuid
<!--NeedCopy-->
```

Query the locking and sharing status of a VDI.

diagnostic-vm-status

```
xe diagnostic-vm-status uuid=vm_uuid
<!--NeedCopy-->
```

Query the hosts on which the VM can boot, check the sharing/locking status of all VBDs.

Disaster recovery commands

Commands for recovering VMs after a disaster

drtask-create

```
xe drtask-create type=type sr-whitelist=sr-white-list device-config=
    device-config
<!--NeedCopy-->
```

Creates a disaster recovery task. For example, to connect to an iSCSI SR in preparation for Disaster Recovery:

```
xe drtask-create type=lvmoiscsi device-config:target=target-ip-address
    \
     device-config:targetIQN=targetIQN device-config:SCSIid=SCSIid \
     sr-whitelist=sr-uuid-list
<!--NeedCopy-->
```

> **Note:**
>
> The command `sr-whitelist` lists SR UUIDs that are allowed. The `drtask-create` command only introduces and connects to an SR which has one of the allowed UUIDs

drtask-destroy

```
xe drtask-destroy uuid=dr-task-uuid
<!--NeedCopy-->
```

Destroys a disaster recovery task and forgets the introduced SR.

vm-assert-can-be-recovered

```
xe vm-assert-can-be-recovered uuid=vm-uuid database:vdi-uuid=vdi-uuid
<!--NeedCopy-->
```

Tests whether storage is available to recover this VM.

appliance-assert-can-be-recovered

```
xe appliance-assert-can-be-recovered uuid=appliance-uuid database:vdi-
    uuid=vdi-uuid
<!--NeedCopy-->
```

Checks whether the storage (containing the appliance's/vAPP disk) is visible.

appliance-recover

```
xe appliance-recover uuid=appliance-uuid database:vdi-uuid=vdi-uuid [
    force=true|false]
<!--NeedCopy-->
```

Recover an appliance/vAPP from the database contained in the supplied VDI.

`vm-recover`

```
xe vm-recover uuid=vm-uuid database:vdi-uuid=vdi-uuid [force=true|false
    ]
<!--NeedCopy-->
```

Recovers a VM from the database contained in the supplied VDI.

`sr-enable-database-replication`

```
xe sr-enable-database-replication uuid=sr_uuid
<!--NeedCopy-->
```

Enables XAPI database replication to the specified (shared) SR.

`sr-disable-database-replication`

```
xe sr-disable-database-replication uuid=sr_uuid
<!--NeedCopy-->
```

Disables XAPI database replication to the specified SR.

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
<!--NeedCopy-->
```

After a disaster, on the secondary site, connect to the SR. The `device-config` command 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
<!--NeedCopy-->
```

Look for database VDIs on the SR:

```
xe vdi-list sr-uuid=sr-uuid type=Metadata
<!--NeedCopy-->
```

Query a database VDI for VMs present:

```
xe vm-list database:vdi-uuid=vdi-uuid
<!--NeedCopy-->
```

Recover a VM:

```
xe vm-recover uuid=vm-uuid database:vdi-uuid=vdi-uuid
<!--NeedCopy-->
```

Destroy the DR task. Any SRs introduced by the DR task and not required by VMs are destroyed:

```
xe drtask-destroy uuid=drtask-uuid
<!--NeedCopy-->
```

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

Class name	Description
vbd	A virtual block device
pbd	The physical block devices through which hosts access SRs

event-wait

```
xe event-wait class=class_name [param-name=param_value] [param-name=/=
    param_value]
<!--NeedCopy-->
```

Blocks other commands from running until an object exists that satisfies the conditions given on the command line. The argument `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
<!--NeedCopy-->
```

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)
<!--NeedCopy-->
```

Blocks other commands until a VM with UUID *$VM* reboots. The command uses the value of `start-time` to decide when the VM reboots.

The class name can be any of the event classes listed at the beginning of this section. The parameters can be any of the parameters 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`. The parameters can be manipulated with the standard parameter commands. For more information, see Low-level parameter commands.

Physical GPU parameters

Physical GPUS (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	Read only
`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 extra configuration parameters for the pGPU	Read/write map parameter
`supported-VGPU-types`	List of virtual GPU types supported by the underlying hardware	Read only
`enabled-VGPU-types`	List of virtual GPU types which have been enabled for this pGPU	Read/Write

Parameter Name	Description	Type
resident-VGPUs	List of vGPUs running on this pGPU	Read only

pgpu-disable-dom0-access

```
xe pgpu-disable-dom0-access uuid=uuid
<!--NeedCopy-->
```

Disable PGPU access to dom0.

pgpu-enable-dom0-access

```
xe pgpu-enable-dom0-access uuid=uuid
<!--NeedCopy-->
```

Enable PGPU access to dom0.

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 virtual GPUs 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

Parameter Name	Description	Type
`other-config`	A list of key/value pairs that specify extra configuration parameters for the GPU group	Read/write map parameter
`supported-VGPU-types`	Union of all virtual GPU types supported by the underlying hardware	Read only
`enabled-VGPU-types`	Union of all virtual GPU types which have been enabled on the underlying pGPUs	Read only
`allocation-algorithm`	Depth-first/Breadth-first setting for allocation virtual GPUs on pGPUs within the group	Read/write enum parameter

GPU group operations Commands for working with GPU Groups

gpu-group-create

```
xe gpu-group-create name-label=name_for_group [name-description=
    description]
<!--NeedCopy-->
```

Creates a new (empty) GPU Group into which pGPUs can be moved.

gpu-group-destroy

```
xe gpu-group-destroy uuid=uuid_of_group
<!--NeedCopy-->
```

Destroys the GPU Group; only permitted for empty groups.

gpu-group-get-remaining-capacity

```
xe gpu-group-get-remaining-capacity uuid=uuid_of_group vgpu-type-uuid=
    uuid_of_vgpu_type
<!--NeedCopy-->
```

Returns how many more virtual GPUs of the specified type can be instantiated in this GPU Group.

gpu-group-param-set

```
xe gpu-group-param-set uuid=uuid_of_group allocation-algorithm=breadth-
    first|depth-first
<!--NeedCopy-->
```

Changes the algorithm that the GPU group uses to allocate virtual GPUs to pGPUs.

Virtual GPU parameters

Virtual GPUs have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the virtual GPU	Read only
`vm-uuid`	The unique identifier/object reference for the VM to which the virtual GPU is assigned	Read only
`vm-name-label`	The name of the VM to which the virtual GPU is assigned	Read only
`gpu-group-uuid`	The unique identifier/object reference for the GPU group in which the virtual GPU is contained	Read only
`gpu-group-name-label`	The name of the GPU group in which the virtual GPU 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 extra configuration parameters for the virtual GPU	Read/write map parameter
`type-uuid`	The unique identifier/object reference for the virtual GPU type of this virtual GPU	Read/write map parameter
`type-model-name`	Model name associated with the virtual GPU type	Read only

Virtual GPU type parameters

> **Note:**
>
> GPU pass-through and virtual GPUs are not compatible with live migration, storage live migration, or VM Suspend unless supported software and graphics cards from GPU vendors are present. VMs without this support cannot be migrated to avoid downtime. For information about NVIDIA vGPU compatibility with live migration, storage live migration, and VM Suspend, see Graphics.

Virtual GPU Types have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the virtual GPU type	Read only
`vendor-name`	Name of virtual GPU vendor	Read only
`model-name`	Model name associated with the virtual GPU type	Read only
`freeze-frame`	Frame buffer size of the virtual GPU type, in bytes	Read only
`max-heads`	Maximum number of displays supported by the virtual GPU type	Read only
`supported-on-PGPUs`	List of pGPUs that support this virtual GPU type	Read only
`enabled-on-PGPUs`	List of pGPUs that have this virtual GPU type enabled	Read only
`VGPU-uuids`	List of virtual GPUs of this type	Read only

Virtual GPU operations

vgpu-create

```
xe vgpu-create vm-uuid=uuid_of_vm gpu_group_uuid=uuid_of_gpu_group [
    vgpu-type-uuid=uuid_of_vgpu-type]
<!--NeedCopy-->
```

Creates a virtual GPU. This command attaches the VM to the specified GPU group and optionally specifies the virtual GPU type. If no virtual GPU type is specified, the 'pass-through'type is assumed.

vgpu-destroy

```
xe vgpu-destroy uuid=uuid_of_vgpu
<!--NeedCopy-->
```

Destroy the specified virtual GPU.

Disabling VNC for VMs with virtual GPU

```
xe vm-param-add uuid=uuid_of_vmparam-name=platform vgpu_vnc_enabled=
    true|false
<!--NeedCopy-->
```

Using **false** disables the VNC console for a VM as it passes disablevnc=1 through to the display emulator. By default, VNC is enabled.

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 commands: xe host-list, xe host-cpu-list, and xe host-crashdump-list). The parameters can be manipulated with the standard parameter commands. For more information, see Low-level parameter commands.

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. You can also specify XenServer by filtering the full list of hosts on the values of fields. For example, specifying enabled=**true** selects all XenServer hosts whose enabled field is equal to **true**. Where multiple XenServer hosts match and the operation can be performed on multiple XenServer hosts, you must specify --multiple to perform the operation. The full list of parameters that can be matched is described at the beginning of this section. You can obtain this list of commands by running the command xe host-list params=all. If no parameters to select XenServer hosts are given, the operation is 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

Parameter Name	Description	Type
`enabled`	Value is **`false`** if disabled. This prevents any new VMs from starting on the hosts and prepares the hosts to be shut down or rebooted. Value is **`true`** if the host is 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 extra 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 host name	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 Change port numbers	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 host name 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 host state may have changed by the time a client reads this field.	Read only set parameter
`current-operations`	Lists the operations currently in process. This list is advisory only and the host state may have changed by the time a client reads this field.	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

Parameter Name	Description	Type
`ha-statefiles`	The UUIDs of all HA state files	Read only
`ha-network-peers`	The UUIDs of all hosts that can host the VMs on this host if there is a failure	Read only
`external-auth-type`	Type of external authentication, for example, Active Directory.	Read only
`external-auth-service-name`	The name of the external authentication service	Read only
`external-auth-configuration`	Configuration information for the external authentication service.	Read only map parameter

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	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 3.00 GHz"	Read only
`stepping`	The CPU revision number	Read only
`flags`	The flags of the physical CPU (a decoded version of the features field)	Read only
`Utilisation`	The current CPU utilization	Read only
`host-uuid`	The UUID if the host the CPU is in	Read only

Parameter Name	Description	Type
`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
`size`	Size of the crashdump, in bytes	Read only

`host-all-editions`

```
xe host-all-editions
<!--NeedCopy-->
```

Get a list of all available editions

`host-apply-edition`

```
xe host-apply-edition [host-uuid=host_uuid] [edition=xenserver_edition=
    "free" "per-socket" "xendesktop"]
<!--NeedCopy-->
```

Assigns the XenServer license to a host server. When you assign a license, XenServer contacts the license server and requests the specified type of license. If a license is available, it is then checked out from the license server.

For initial licensing configuration, see also `license-server-address` and `license-server-port`.

`host-backup`

```
xe host-backup file-name=backup_filename host=host_name
<!--NeedCopy-->
```

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

> **Important:**
>
> While the `xe host-backup` command works if run on the local host (that is, without a specific host name specified), do not use it this way. Doing so would fill up the control domain partition with the backup file. Only use the command from a remote off-host machine where you have space to hold the backup file.

`host-bugreport-upload`

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

Generate a fresh bug report (using `xen-bugtool`, with all optional files included) and upload to the Support FTP site or some other location.

The hosts on which to perform this operation 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 is the default Support FTP site.

`host-call-plugin`

```
xe host-call-plugin host-uuid=host_uuid plugin=plugin fn=function [args
    =args]
<!--NeedCopy-->
```

Calls the function within the plug-in on the given host with optional arguments.

host-compute-free-memory

```
xe host-compute-free-memory
<!--NeedCopy-->
```

Computes the amount of free memory on the host.

host-compute-memory-overhead

```
xe host-compute-memory-overhead
<!--NeedCopy-->
```

Computes the virtualization memory overhead of a host.

host-cpu-info

```
xe host-cpu-info [uuid=uuid]
<!--NeedCopy-->
```

Lists information about the host's physical CPUs.

host-crashdump-destroy

```
xe host-crashdump-destroy uuid=crashdump_uuid
<!--NeedCopy-->
```

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

host-crashdump-upload

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

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

host-declare-dead

```
xe host-declare-dead uuid=host_uuid
<!--NeedCopy-->
```

Declare that the host is dead without contacting it explicitly.

> **Warning:**
>
> This call is dangerous and can cause data loss if the host is not actually dead.

host-disable

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

Disables the specified XenServer hosts, which prevents any new VMs from starting on them. This action prepares the XenServer hosts to be shut down or rebooted. After that host reboots, if all conditions for enabling are met (for example, storage is available), the host is automatically re-enabled.

The hosts on which to perform this operation are selected 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.

host-disable-display

```
xe host-disable-display uuid=host_uuid
<!--NeedCopy-->
```

Disable display for the host.

host-disable-local-storage-caching

```
xe host-disable-local-storage-caching
<!--NeedCopy-->
```

Disable local storage caching on the specified host.

host-dmesg

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

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

The hosts on which to perform this operation 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-ha-disable`

```
xe host-emergency-ha-disable  [--force]
<!--NeedCopy-->
```

Disable HA on the local host. Only to be used to recover a pool with a broken HA setup.

`host-emergency-management-reconfigure`

```
xe host-emergency-management-reconfigure interface=
    uuid_of_management_interface_pif
<!--NeedCopy-->
```

Reconfigure the management interface of this XenServer host. Use this command only if the XenServer host is in emergency mode. Emergency mode means that the host is a member in a resource pool whose pool coordinator has disappeared from the network and cannot be contacted after a number of retries.

`host-emergency-reset-server-certificate`

```
xe host-emergency-reset-server-certificate
<!--NeedCopy-->
```

Installs a self-signed certificate on the XenServer host where the command is run.

`host-enable`

```
xe host-enable [host-selector=host_selector_value...]
<!--NeedCopy-->
```

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

The hosts on which to perform this operation 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-enable-display`

```
xe host-enable-display uuid=host_uuid
<!--NeedCopy-->
```

Enable display for the host.

host-enable-local-storage-caching

```
xe host-enable-local-storage-caching  sr-uuid=sr_uuid
<!--NeedCopy-->
```

Enable local storage caching on the specified host.

host-evacuate

```
xe host-evacuate [host-selector=host_selector_value...]
<!--NeedCopy-->
```

Live migrates all running VMs to other suitable hosts on a pool. First, disable the host by using the `host-disable` command.

If the evacuated host is the pool coordinator, then another host must be selected to be the pool coordinator. To change the pool coordinator with HA disabled, use the `pool-designate-new-master` command. For more information, see pool-designate-new-master.

With HA enabled, your only option is to shut down the XenServer host, which causes HA to elect a new pool coordinator at random. For more information, see host-shutdown.

The hosts on which to perform this operation 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

```
xe host-forget uuid=host_uuid
<!--NeedCopy-->
```

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.

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

`host-get-server-certificate`

```
xe host-get-server-certificate
<!--NeedCopy-->
```

Get the installed server TLS certificate.

`host-get-sm-diagnostics`

```
xe host-get-sm-diagnostics uuid=uuid
<!--NeedCopy-->
```

Display per-host SM diagnostic information.

`host-get-system-status`

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

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. For more information, see host-get-system-status-capabilities. 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 hosts on which to perform this operation are selected using the standard selection mechanism (see host selectors above).

`host-get-system-status-capabilities`

```
xe host-get-system-status-capabilities [host-selector=
    host_selector_value...]
<!--NeedCopy-->
```

Get system status capabilities for the specified hosts. The capabilities are returned as an XML fragment that similar to the following example:

```
<?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>
<!--NeedCopy-->
```

Each capability entity can have the following attributes.

- `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 selects 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 that the time is unimportant.
- `pii` Personally identifiable information. Indicates whether the entry has information that can identify the system owner or details of their network topology. The attribute can have one of the following values:
  - `no`: no PII is in these entries
  - `yes`: PII likely or certainly is in these entries
  - `maybe`: you might want to audit these entries for PII
  - `if_customized` if the files are unmodified, then they contain no PII. However, because we encourage editing of these files, PII might have been introduced by such customization. This value 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 hosts on which to perform this operation are selected using the standard selection mechanism (see host selectors above).

`host-get-thread-diagnostics`

```
xe host-get-thread-diagnostics uuid=uuid
<!--NeedCopy-->
```

Display per-host thread diagnostic information.

`host-get-vms-which-prevent-evacuation`

```
xe host-get-vms-which-prevent-evacuation uuid=uuid
<!--NeedCopy-->
```

Return a list of VMs which prevent the evacuation of a specific host and display reasons for each one.

`host-is-in-emergency-mode`

```
xe host-is-in-emergency-mode
<!--NeedCopy-->
```

Returns **`true`** if the host the CLI is talking to is in emergency mode, **`false`** otherwise. This CLI command works directly on pool member hosts even with no pool coordinator present.

`host-license-add`

```
xe host-license-add [license-file=path/license_filename] [host-uuid=
    host_uuid]
<!--NeedCopy-->
```

For XenServer, use to parse a local license file and add it to the specified XenServer host.

`host-license-remove`

```
xe host-license-remove [host-uuid=host_uuid]
<!--NeedCopy-->
```

Remove any licensing applied to a host.

`host-license-view`

```
xe host-license-view [host-uuid=host_uuid]
<!--NeedCopy-->
```

Displays the contents of the XenServer host license.

host-logs-download

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

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 file name using the optional parameter *file-name*.

The hosts on which to perform this operation 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.

> **Important:**
>
> While the `xe host-logs-download` command works if run on the local host (that is, without a specific host name specified), do *not* use it this way. Doing so clutters the control domain partition with the copy of the logs. Only use the command from a remote off-host machine where you have space to hold the copy of the logs.

host-management-disable

```
xe host-management-disable
<!--NeedCopy-->
```

Disables the host agent listening on an external management network interface and disconnects all connected API clients (such as the XenCenter). This command operates directly on the XenServer host the CLI is connected to. The command is not forwarded to the pool coordinator when applied to a member XenServer host.

> **Warning:**
>
> Be careful when using this CLI command off-host. After this command is run, you cannot connect to the control domain remotely over the network to re-enable the host agent.

host-management-reconfigure

```
xe host-management-reconfigure [interface=device] [pif-uuid=uuid]
<!--NeedCopy-->
```

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

If the device name of an interface (which must have an IP address) is specified, the XenServer host immediately rebinds. This command 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 run.

> **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 are unable to reach the XenServer host.

`host-power-on`

```
xe host-power-on [host=host_uuid]
<!--NeedCopy-->
```

Turns on power on XenServer hosts with the *Host Power On* function enabled. Before using this command, enable `host-set-power-on` on the host.

`host-reboot`

```
xe host-reboot [host-selector=host_selector_value...]
<!--NeedCopy-->
```

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 hosts on which to perform this operation 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 is handled and the pool recovers when the XenServer hosts return. The other members and the pool coordinator continue to function.

If you shut down the pool coordinator, the pool is out of action until one of the following actions occurs:

- You make one of the members into the pool coordinator
- The original pool coordinator is rebooted and back on line.

  When the pool coordinator is back online, the members reconnect and synchronize with the pool coordinator.

host-restore

```
xe host-restore [file-name=backup_filename] [host-selector=
    host_selector_value...]
<!--NeedCopy-->
```

Restore a backup named `file-name` of the XenServer host control software. 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 hosts on which to perform this operation 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-send-debug-keys

```
xe host-send-debug-keys  host-uuid=host_uuid keys=keys
<!--NeedCopy-->
```

Send specified hypervisor debug keys to specified host.

host-server-certificate-install

```
xe host-server-certificate-install certificate=path_to_certificate_file
     private-key=path_to_private_key [certificate-chain=
    path_to_chain_file] [host=host_name | uuid=host_uuid]
<!--NeedCopy-->
```

Install a TLS certificate on a XenServer host.

host-set-hostname-live

```
xe host-set-hostname-live host-uuid=uuid_of_host host-name=new_hostname
<!--NeedCopy-->
```

Change the host name of the XenServer host specified by `host-uuid`. This command persistently sets both the host name in the control domain database and the actual Linux host name of the XenServer host. The value of `host-name` is *not* the same as the value of the `name_label` field.

host-set-power-on-mode

```
xe host-set-power-on-mode host=host_uuid power-on-mode={
 "" | "wake-on-lan" | "DRAC" | "custom" }
   \
     [ power-on-config:power_on_ip=ip-address power-on-config:
        power_on_user=user power-on-config:power_on_password_secret=
        secret-uuid ]
<!--NeedCopy-->
```

Use to enable the *Host Power On* function on XenServer hosts that are compatible with remote power solutions. 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"`. For more information, see Secrets.

`host-shutdown`

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

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 hosts on which to perform this operation 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 is handled and the pool recovers when the XenServer hosts returns. The other members and the pool coordinator continue to function.

If you shut down the pool coordinator, the pool is out of action until one of the following actions occurs:

- You make one of the members into the pool coordinator
- The original pool coordinator is rebooted and back on line.

  When the pool coordinator is back online, the members reconnect and synchronize with the pool coordinator.

If HA is enabled for the pool, one of the members is made into a pool coordinator automatically. If HA is disabled, you must manually designate the desired XenServer host as pool coordinator with the `pool-designate-new-master` command. For more information, see pool-designate-new-master.

host-sm-dp-destroy

```
xe host-sm-dp-destroy uuid=uuid dp=dp [allow-leak=true|false]
<!--NeedCopy-->
```

Attempt to destroy and clean up a storage datapath on a host. If `allow-leak=true` is provided then it deletes all records of the datapath even if it is not shut down cleanly.

host-sync-data

```
xe host-sync-data
<!--NeedCopy-->
```

Synchronize the data stored on the pool coordinator with the named host. This does not include the database data).

host-syslog-reconfigure

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

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

The hosts on which to perform this operation 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

```
xe host-data-source-list [host-selectors=host selector value...]
<!--NeedCopy-->
```

List the data sources that can be recorded for a host.

Select the hosts 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 is performed on all hosts.

Data sources have two parameters –`standard` and `enabled`. This command outputs the values of the parameters:

- If a data source has `enabled` set to `true`, the metrics are currently being recorded to the performance database.

- If a data source has `standard` set to **`true`**, the metrics are recorded to the performance database *by default*. The value of `enabled` is also set to **`true`** for this data source.
- If a data source has `standard` set to **`false`**, the metrics are *not* recorded to the performance database by default. The value of `enabled` is also 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 command sets `enabled` to **`true`**. To stop, run the `host-data-source-forget`. This command sets `enabled` to **`false`**.

host-data-source-record

```
xe host-data-source-record data-source=name_description_of_data_source
    [host-selectors=host_selector_value...]
<!--NeedCopy-->
```

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 hosts. For performance reasons, this database is distinct from the normal agent database.

Select the hosts 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 is performed on all hosts.

host-data-source-forget

```
xe host-data-source-forget data-source=name_description_of_data_source
    [host-selectors=host_selector_value...]
<!--NeedCopy-->
```

Stop recording the specified data source for a host and forget all of the recorded data.

Select the hosts 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 is performed on all hosts.

host-data-source-query

```
xe host-data-source-query data-source=name_description_of_data_source [
    host-selectors=host_selector_value...]
<!--NeedCopy-->
```

Display the specified data source for a host.

Select the hosts 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 is performed on all hosts.

Message commands

Commands for working with messages. Messages are created to notify users of significant events, and are displayed in XenCenter as alerts.

The message objects can be listed with the standard object listing command (`xe message-list`), and the parameters manipulated with the standard parameter commands. For more information, see Low-level parameter commands

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`

```
xe 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]]
<!--NeedCopy-->
```

Creates a message.

`message-destroy`

```
xe message-destroy [uuid=message_uuid]
<!--NeedCopy-->
```

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
<!--NeedCopy-->
```

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. For more information, see Low-level parameter commands

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

Parameter Name	Description	Type
`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
`default-locking-mode`	A network object used with VIF objects for ARP filtering. Set to `unlocked` to remove all the filtering rules associated with the VIF. Set to `disabled` so the VIF drops all traffic.	Read/write
`purpose`	Set of purposes for which the XenServer host uses this network. Set to `nbd` to use the network to make NBD connections.	Read/write
`other-config:staticroutes`	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 172.16.0.0/15/192.168.0.3,172.18.0.0/16/192.168.0.4 causes traffic on 172.16.0.0/15 to be routed over 192.168.0.3 and traffic on 172.18.0.0/16 to be routed over 192.168.0.4.	Read/write
`other-config:ethtoolautoneg`	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

Parameter Name	Description	Type
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

```
xe network-create name-label=name_for_network [name-description=
    descriptive_text]
<!--NeedCopy-->
```

Creates a network.

network-destroy

```
xe network-destroy uuid=network_uuid
<!--NeedCopy-->
```

Destroys an existing network.

SR-IOV commands

Commands for working with SR-IOV.

The network-sriov objects can be listed with the standard object listing command (xe network-sriov-list), and the parameters manipulated with the standard parameter commands. For more information, see Low-level parameter commands

SR-IOV parameters

SR-IOV has the following parameters:

Parameter Name	Description	Type
`physical-PIF`	The PIF to enable SR-IOV.	Read only
`logical-PIF`	An SR-IOV logical PIF. Users can use this parameter to create an SR-IOV VLAN network.	Read only
`requires-reboot`	If set to True, used to reboot host to bring SR-IOV enabling into effect.	Read only
`remaining-capacity`	Number of available VFs remaining.	Read only

`network-sriov-create`

```
xe network-sriov-create network-uuid=network_uuid pif-uuid=
    physical_pif_uuid
<!--NeedCopy-->
```

Creates an SR-IOV network object for a given physical PIF and enables SR-IOV on the physical PIF.

`network-sriov-destroy`

```
xe network-sriov-destroy uuid=network_sriov_uuid
<!--NeedCopy-->
```

Removes a network SR-IOV object and disables SR-IOV on its physical PIF.

Assign an SR-IOV VF

```
xe vif-create device=device_index mac=vf_mac_address network-uuid=
    sriov_network vm-uuid=vm_uuid
<!--NeedCopy-->
```

Assigns a VF from an SR-IOV network to a VM.

SDN Controller commands

Commands for working with the SDN controller.

`sdn-controller-forget`

```
xe sdn-controller-introduce [address=address] [protocol=protocol] [tcp-
    port=tcp_port]
<!--NeedCopy-->
```

Introduce an SDN controller.

`sdn-controller-introduce`

```
xe sdn-controller-forget uuid=uuid
<!--NeedCopy-->
```

Remove an SDN controller.

Tunnel commands

Commands for working with tunnels.

`tunnel-create`

```
xe tunnel-create pif-uuid=pif_uuid network-uuid=network_uuid
<!--NeedCopy-->
```

Create a new tunnel on a host.

`tunnel-destroy`

```
xe tunnel-destroy uuid=uuid
<!--NeedCopy-->
```

Destroy a tunnel.

Patch commands

Commands for working with patches.

`patch-apply`

```
xe patch-apply uuid=patch_uuid host-uuid=host_uuid
<!--NeedCopy-->
```

Apply the previously uploaded patch to the specified host.

patch-clean

```
xe patch-clean uuid=uuid
<!--NeedCopy-->
```

Delete a previously uploaded patch file.

patch-destroy

```
xe patch-destroy uuid=uuid
<!--NeedCopy-->
```

Remove an unapplied patch record and files from the host.

patch-pool-apply

```
xe patch-pool-apply uuid=uuid
<!--NeedCopy-->
```

Apply the previously uploaded patch to all hosts in the pool.

patch-pool-clean

```
xe patch-pool-clean uuid=uuid
<!--NeedCopy-->
```

Delete a previously uploaded patch file on all hosts in the pool.

patch-precheck

```
xe patch-precheck uuid=uuid host-uuid=host_uuid
<!--NeedCopy-->
```

Run the prechecks contained within the patch previously uploaded to the specified host.

patch-upload

```
xe patch-upload file-name=file_name
<!--NeedCopy-->
```

Upload a patch file to the host.

PBD commands

Commands for working with PBDs (Physical Block Devices). PBDs 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. For more information, see Low-level parameter commands

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`	Extra configuration information that is provided to the SR-backend-driver of a host	Read only map parameter
`currently-attached`	True if the SR is 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`	Extra configuration information.	Read/write map parameter

`pbd-create`

```
xe pbd-create host-uuid=uuid_of_host sr-uuid=uuid_of_sr [device-config:
    key=corresponding_value]
<!--NeedCopy-->
```

Create a PBD on your XenServer host. The read-only `device-config` parameter can only be set on creation.

To add a mapping from 'path'to '/tmp', ensure that the command line contains 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`

```
xe pbd-destroy uuid=uuid_of_pbd
<!--NeedCopy-->
```

Destroy the specified PBD.

`pbd-plug`

```
xe pbd-plug uuid=uuid_of_pbd
<!--NeedCopy-->
```

Attempts to plug in the PBD to the XenServer host. If this command succeeds, the referenced SR (and the VDIs contained within) becomes visible to the XenServer host.

`pbd-unplug`

```
xe pbd-unplug uuid=uuid_of_pbd
<!--NeedCopy-->
```

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. For more information, see Low-level parameter commands

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`	Extra 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 that no VLAN tag is assigned	Read only
`bond-master-of`	The UUID of the bond this PIF is the main interface of (if any)	Read only
`bond-slave-of`	The UUID of the bond this PIF is part 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 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

Parameter Name	Description	Type
`IP-configuration-mode`	Type of network address configuration used; DHCP or static	Read only
`IP`	IP address of the PIF. Defined here when IP-configuration-mode is static; undefined when DHCP	Read only
`netmask`	Netmask of the PIF. Defined here when IP-configuration-mode is static; undefined when supplied by DHCP	Read only
`gateway`	Gateway address of the PIF. Defined here when IP-configuration-mode is static; undefined when supplied by DHCP	Read only
`DNS`	DNS address of the PIF. Defined here when IP-configuration-mode is static; undefined when 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

Parameter Name	Description	Type
`other-config:ethtoolspeed`	Sets the speed of connection in Mbps	Read/write
`other-config:ethtoolautoneg`	Set to no to disable autonegotiation of the physical interface or bridge. Default is yes.	Read/write
`other-config:ethtoolduplex`	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
`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:bondmiimon`	Interval between link liveness checks, in milliseconds	Read/write
`other-config:bonddowndelay`	Number of milliseconds to wait after link is lost before really considering the link to have gone. This parameter allows for transient link loss	Read/write

Parameter Name	Description	Type
`other-config:bondupdelay`	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`

```
xe pif-forget uuid=uuid_of_pif
<!--NeedCopy-->
```

Destroy the specified PIF object on a particular host.

`pif-introduce`

```
xe pif-introduce host-uuid=host_uuid mac=mac_address_for_pif device=
    interface_name
<!--NeedCopy-->
```

Create a PIF object representing a physical interface on the specified XenServer host.

`pif-plug`

```
xe pif-plug uuid=uuid_of_pif
<!--NeedCopy-->
```

Attempt to bring up the specified physical interface.

pif-reconfigure-ip

```
xe 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]
<!--NeedCopy-->
```

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-reconfigure-ipv6

```
xe pif-reconfigure-ipv6 uuid=uuid_of_pif mode=mode [gateway=
    network_gateway_address] [IPv6=static_ip_for_this_pif] [DNS=
    dns_address]
<!--NeedCopy-->
```

Reconfigure the IPv6 address settings on a PIF.

pif-scan

```
xe pif-scan host-uuid=host_uuid
<!--NeedCopy-->
```

Scan for new physical interfaces on your XenServer host.

pif-set-primary-address-type

```
xe pif-set-primary-address-type  uuid=uuid primary_address_type=
    address_type
<!--NeedCopy-->
```

Change the primary address type used by this PIF.

pif-unplug

```
xe pif-unplug uuid=uuid_of_pif
<!--NeedCopy-->
```

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 host in the pool can be migrated in near-real time to another host in the pool. This migration happens while the VM is still running, without it needing to be shut down and brought back up.

Each XenServer host is really a pool consisting of a single member by default. When your XenServer host is joined to a pool, it is designated as a member. If the pool that the host is joined to consists of a single member, that member becomes the pool coordinator. If the pool that the host is joined to already has multiple members, one of these members is already the pool coordinator and remains so when the new host joins the pool.

The singleton pool object can be listed with the standard object listing command (`xe pool-list`). Its parameters can be manipulated with the standard parameter commands. For more information, see Low-level parameter commands

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 coordinator	Read only
`default-SR`	The unique identifier/object reference of the default SR for the pool	Read/write

Parameter Name	Description	Type
`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
`other-config`	A list of key/value pairs that specify extra configuration parameters for the pool	Read/write map parameter
`other-config:default_ha_timeout`	High availability timeout in seconds.	Read/write
`supported-sr-types`	SR types that this pool can use	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 overcommitted	Read only
`blobs`	Binary data store	Read only

Parameter Name	Description	Type
`live-patching-disabled`	Set to False to enable live patching. Set to True to disable live patching.	Read/write
`igmp-snooping-enabled`	Set to True to enable IGMP snooping. Set to False to disable IGMP snooping.	Read/write
`https-only`	Set to False to allow external clients that use the management API to connect to XenServer using either HTTPS over port 443 or HTTP over port 80. Set to True to block port 80 and require clients to exclusively connect using HTTPS over port 443.	Read/write
`migration-compression`	Set to True to enable migration stream compression for your XenServer pool. Set to False to disable migration stream compression. The default is False. Can be overridden by `vm-migrate` command's `compress` parameter.	Read/write

`pool-apply-edition`

```
xe pool-apply-edition edition=edition [uuid=uuid] [license-server-
    address=address] [license-server-port=port]
<!--NeedCopy-->
```

Apply an edition across the pool.

`pool-certificate-install`

```
xe pool-certificate-install filename=file_name
<!--NeedCopy-->
```

Install an TLS certificate, pool-wide.

pool-certificate-list

```
xe pool-certificate-list
<!--NeedCopy-->
```

List all installed TLS certificates in a pool.

pool-certificate-sync

```
xe pool-certificate-sync
<!--NeedCopy-->
```

Sync TLS certificates and certificate revocation lists from pool coordinator to the other pool members.

pool-certificate-uninstall

```
xe pool-certificate-uninstall name=name
<!--NeedCopy-->
```

Uninstall a TLS certificate.

pool-crl-install

```
xe pool-crl-install filename=file_name
<!--NeedCopy-->
```

Install a TLS certificate revocation list, pool-wide.

pool-crl-list

```
xe pool-crl-list
<!--NeedCopy-->
```

List all installed TLS certificate revocation lists.

pool-crl-uninstall

```
xe pool-crl-uninstall name=name
<!--NeedCopy-->
```

Uninstall an TLS certificate revocation list.

pool-deconfigure-wlb

```
xe pool-deconfigure-wlb
<!--NeedCopy-->
```

Permanently remove the configuration for workload balancing.

pool-designate-new-master

```
xe pool-designate-new-master host-uuid=uuid_of_new_master
<!--NeedCopy-->
```

Instruct the specified member XenServer host to become the coordinator (formerly called "master") of an existing pool. This command performs an orderly handover of the role of pool coordinator to another host in the resource pool. This command only works when the current pool coordinator is online. It is not a replacement for the emergency mode commands listed below.

pool-disable-external-auth

```
xe pool-disable-external-auth [uuid=uuid] [config=config]
<!--NeedCopy-->
```

Disables external authentication in all the hosts in a pool.

pool-disable-local-storage-caching

```
xe pool-disable-local-storage-caching uuid=uuid
<!--NeedCopy-->
```

Disable local storage caching across the pool.

pool-disable-redo-log

```
xe pool-disable-redo-log
<!--NeedCopy-->
```

Disable the redo log if in use, unless HA is enabled.

pool-dump-database

```
xe pool-dump-database file-name=filename_to_dump_database_into_(
    on_client)
<!--NeedCopy-->
```

Download a copy of the entire pool database and dump it into a file on the client.

pool-enable-external-auth

```
xe pool-enable-external-auth  auth-type=auth_type service-name=
    service_name [uuid=uuid] [config:=config]
<!--NeedCopy-->
```

Enables external authentication in all the hosts in a pool. Note that some values of `auth-type` will require particular `config:` values.

pool-enable-local-storage-caching

```
xe pool-enable-local-storage-caching uuid=uuid
<!--NeedCopy-->
```

Enable local storage caching across the pool.

pool-enable-redo-log

```
xe pool-ensable-redo-log sr-uuid=sr_uuid
<!--NeedCopy-->
```

Enable the redo log on the given SR if in use, unless HA is enabled.

pool-eject

```
xe pool-eject host-uuid=uuid_of_host_to_eject
<!--NeedCopy-->
```

Instruct the specified XenServer host to leave an existing pool.

pool-emergency-reset-master

```
xe pool-emergency-reset-master master-address=address_of_pool_master
<!--NeedCopy-->
```

Instruct a pool member host to reset its pool coordinator address to the new value and attempt to connect to it. Do not run this command on pool coordinators.

`pool-emergency-transition-to-master`

```
xe pool-emergency-transition-to-master
<!--NeedCopy-->
```

Instruct a member XenServer host to become the pool coordinator (formerly called "the pool master"). The XenServer host accepts this command only after the host has transitioned to emergency mode. Emergency mode means it is a member of a pool whose coordinator has disappeared from the network and cannot be contacted after some number of retries.

If the host password has been modified since the host joined the pool, this command can cause the password of the host to reset. For more information, see (User commands).

`pool-ha-enable`

```
xe pool-ha-enable heartbeat-sr-uuids=uuid_of_heartbeat_sr
<!--NeedCopy-->
```

Enable high availability on the resource pool, using the specified SR UUID as the central storage heartbeat repository.

`pool-ha-disable`

```
xe pool-ha-disable
<!--NeedCopy-->
```

Disables the high availability feature on the resource pool.

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

Compute the maximum number of host failures to tolerate under the current pool configuration.

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

```
xe pool-ha-compute-hypothetical-max-host-failures-to-tolerate [vm-uuid=
    vm_uuid] [restart-priority=restart_priority]
<!--NeedCopy-->
```

Compute the maximum number of host failures to tolerate with the supplied, proposed protected VMs.

`pool-initialize-wlb`

```
xe pool-initialize-wlb wlb_url=url wlb_username=wlb_username
    wlb_password=wlb_password xenserver_username=username
    xenserver_password=password
<!--NeedCopy-->
```

Initialize workload balancing for the current pool with the target Workload Balancing server.

`pool-join`

```
xe pool-join master-address=address master-username=username master-
    password=password
<!--NeedCopy-->
```

Instruct your XenServer host to join an existing pool.

`pool-management-reconfigure`

```
xe pool-management-reconfigure [network-uuid=network-uuid]
<!--NeedCopy-->
```

Reconfigures the management interface of all the hosts in the pool to use the specified network interface, which is the interface that is used to connect to the XenCenter. The command rewrites the MANAGEMENT_INTERFACE key in `/etc/xensource-inventory` for all the hosts in the pool.

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

From the network UUID specified, UUID of the PIF object is identified and mapped to the XenServer host, which determines which IP address to rebind to itself. It must not be in emergency mode when this command is run.

> **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 are unable to reach the XenServer host.

`pool-recover-slaves`

```
xe pool-recover-slaves
<!--NeedCopy-->
```

Instruct the pool coordinator to try to reset the address of all members currently running in emergency mode. This command is typically used after `pool-emergency-transition-to-master` has been used to set one of the members as the new pool coordinator.

pool-restore-database

```
xe pool-restore-database file-name=filename_to_restore_from_on_client [
    dry-run=true|false]
<!--NeedCopy-->
```

Upload a database backup (created with `pool-dump-database`) to a pool. On receiving the upload, the pool coordinator restarts 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-retrieve-wlb-configuration

```
xe pool-retrieve-wlb-configuration
<!--NeedCopy-->
```

Retrieves the pool optimization criteria from the Workload Balancing server.

pool-retrieve-wlb-diagnostics

```
xe pool-retrieve-wlb-diagnostics [filename=file_name]
<!--NeedCopy-->
```

Retrieves diagnostics from the Workload Balancing server.

pool-retrieve-wlb-recommendations

```
xe pool-retrieve-wlb-recommendations
<!--NeedCopy-->
```

Retrieves VM migrate recommendations for the pool from the Workload Balancing server.

pool-retrieve-wlb-report

```
xe pool-retrieve-wlb-report report=report [filename=file_name]
<!--NeedCopy-->
```

Retrieves reports from the Workload Balancing server.

`pool-secret-rotate`

```
xe pool-secret-rotate
<!--NeedCopy-->
```

Rotate the pool secret.

The pool secret is a secret shared among the XenServer hosts in a pool that enables the host to prove its membership to a pool. Users with the Pool Admin role can view this secret when connecting to the host over SSH. Rotate the pool secret if one of these users leaves your organization or loses their Pool Admin role.

`pool-send-test-post`

```
xe pool-send-test-post dest-host=destination_host dest-port=
    destination_port body=post_body
<!--NeedCopy-->
```

Send the given body to the given host and port, using HTTPS, and print the response. This is used for debugging the TLS layer.

`pool-send-wlb-configuration`

```
xe pool-send-wlb-configuration [config:=config]
<!--NeedCopy-->
```

Sets the pool optimization criteria for the Workload Balancing server.

`pool-sync-database`

```
xe pool-sync-database
<!--NeedCopy-->
```

Force the pool database to be synchronized across all hosts in the resource pool. This command is not necessary in normal operation since the database is regularly automatically replicated. However, the command can be useful for ensuring changes are rapidly replicated after performing a significant set of CLI operations.

Set `https-only`

```
xe pool-param-set [uuid=pool-uuid] [https-only=true|false]
<!--NeedCopy-->
```

Enables or disables the blocking of port 80 on the management interface of XenServer hosts.

PVS Accelerator commands

Commands for working with the PVS Accelerator.

`pvs-cache-storage-create`

```
xe pvs-cache-storage-create sr-uuid=sr_uuid pvs-site-uuid=pvs_site_uuid
    size=size
<!--NeedCopy-->
```

Configure a PVS cache on a given SR for a given host.

`pvs-cache-storage-destroy`

```
xe pvs-cache-storage-destroy uuid=uuid
<!--NeedCopy-->
```

Remove a PVS cache.

`pvs-proxy-create`

```
xe pvs-proxy-create pvs-site-uuid=pvs_site_uuid vif-uuid=vif_uuid
<!--NeedCopy-->
```

Configure a VM/VIF to use a PVS proxy.

`pvs-proxy-destroy`

```
xe pvs-proxy-destroy uuid=uuid
<!--NeedCopy-->
```

Remove (or switch off) a PVS proxy for this VIF/VM.

`pvs-server-forget`

```
xe pvs-server-forget uuid=uuid
<!--NeedCopy-->
```

Forget a PVS server.

`pvs-server-introduce`

```
xe pvs-server-introduce addresses=adresses first-port=first_port last-
    port=last_port pvs-site-uuid=pvs_site_uuid
<!--NeedCopy-->
```

Introduce new PVS server.

`pvs-site-forget`

```
xe pvs-site-forget uuid=uuid
<!--NeedCopy-->
```

Forget a PVS site.

`pvs-site-introduce`

```
xe pvs-site-introduce name-label=name_label [name-description=
    name_description] [pvs-uuid=pvs_uuid]
<!--NeedCopy-->
```

Introduce new PVS site.

Storage Manager commands

Commands for controlling Storage Manager plug-ins.

The storage manager objects can be listed with the standard object listing command (`xe sm-list`). The parameters can be manipulated with the standard parameter commands. For more information, see Low-level parameter commands

SM parameters

SMs have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the SM plug-in	Read only
`name-label`	The name of the SM plug-in	Read only

Parameter Name	Description	Type
`name-description`	The description string of the SM plug-in	Read only
`type`	The SR type that this plug-in connects to	Read only
`vendor`	Name of the vendor who created this plug-in	Read only
`copyright`	Copyright statement for this SM plug-in	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 plug-in	Read only
`driver-filename`	The file name of the SR driver.	Read only

Snapshot commands

Commands for working with snapshots.

`snapshot-clone`

```
xe snapshot-clone new-name-label=name_label [uuid=uuid] [new-name-
    description=description]
<!--NeedCopy-->
```

Create a new template by cloning an existing snapshot, using storage-level fast disk clone operation where available.

`snapshot-copy`

```
xe snapshot-copy new-name-label=name_label [uuid=uuid] [new-name-
    description=name_description] [sr-uuid=sr_uuid]
<!--NeedCopy-->
```

Create a new template by copying 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' - i.e. not part of a CoW chain.

snapshot-destroy

```
xe snapshot-destroy  [uuid=uuid] [snapshot-uuid=snapshot_uuid]
<!--NeedCopy-->
```

Destroy a snapshot. This leaves the storage associated with the snapshot intact. To delete storage too, use snapshot-uninstall.

snapshot-disk-list

```
xe snapshot-disk-list [uuid=uuid] [snapshot-uuid=snapshot_uuid] [vbd-
    params=vbd_params] [vdi-params=vdi_params]
<!--NeedCopy-->
```

List the disks on the selected VM(s).

snapshot-export-to-template

```
xe snapshot-export-to-template filename=file_name snapshot-uuid=
    snapshot_uuid  [preserve-power-state=true|false]
<!--NeedCopy-->
```

Export a snapshot to *file name*.

snapshot-reset-powerstate

```
xe snapshot-reset-powerstate [uuid=uuid] [snapshot-uuid=snapshot_uuid]
    [--force]
<!--NeedCopy-->
```

Force the VM power state to halted in the management toolstack database only. This command is used to recover a snapshot that is marked as 'suspended'. This is a potentially dangerous operation: you must ensure that you do not need the memory image anymore. You will not be able to resume your snapshot anymore.

snapshot-revert

```
xe snapshot-revert [uuid=uuid] [snapshot-uuid=snapshot_uuid]
<!--NeedCopy-->
```

Revert an existing VM to a previous checkpointed or snapshot state.

snapshot-uninstall

```
xe snapshot-uninstall [uuid=uuid] [snapshot-uuid=snapshot_uuid] [--
    force]
<!--NeedCopy-->
```

Uninstall a snapshot. This operation will destroy those VDIs that are marked RW and connected to this snapshot only. To simply destroy the VM record, use snapshot-destroy.

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. For more information, see Low-level parameter commands

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
`host`	The storage repository host name	Read only
`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

Parameter Name	Description	Type
`virtual-allocation`	Sum of virtual-size values of all VDIs in this storage repository (in bytes)	Read only
`physical-utilisation`	Physical space currently utilized on this SR, in bytes. For thin provisioned 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 back-end driver to use	Read only
`content-type`	The type of the SR's content. Used to distinguish ISO libraries from other SRs. For storage repositories that store a library of ISOs, the `content-type` must be set to `iso`. In other cases, we recommend that you set this parameter either to empty, or the string user.	Read only
`shared`	True if this SR can be shared between multiple hosts. False otherwise.	Read/write
`introduced-by`	The `drtask` (if any) which introduced the SR	Read only
`is-tools-sr`	True if this is the SR that contains the Tools ISO VDIs. False otherwise.	Read only
`other-config`	List of key/value pairs that specify extra configuration parameters for the SR	Read/write map parameter
`sm-config`	SM dependent data	Read only map parameter
`blobs`	Binary data store	Read only

Parameter Name	Description	Type
`local-cache-enabled`	True if this SR is assigned to be the local cache for its host. False otherwise.	Read only
`tags`	User-specified tags for categorization purposes	Read/write set parameter
`clustered`	True it the SR is using aggregated local storage. False otherwise.	Read only

sr-create

```
xe sr-create name-label=name physical-size=size type=type content-type=
    content_type device-config:config_name=value [host-uuid=host_uuid] [
    shared=true|false]
<!--NeedCopy-->
```

Creates an SR on the disk, introduces it into the database, and creates a PBD attaching the SR to the 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`. For details of these parameters across the different storage back-ends, see Create an SR.

sr-data-source-forget

```
xe sr-data-source-forget data-source=data_source
<!--NeedCopy-->
```

Stop recording the specified data source for a SR, and forget all of the recorded data.

sr-data-source-list

```
xe sr-data-source-list
<!--NeedCopy-->
```

List the data sources that can be recorded for a SR.

sr-data-source-query

```
xe sr-data-source-query data-source=data_source
<!--NeedCopy-->
```

Query the last value read from a SR data source.

sr-data-source-record

```
xe sr-data-source-record  data-source=data_source
<!--NeedCopy-->
```

Record the specified data source for a SR.

sr-destroy

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

Destroys the specified SR on the XenServer host.

sr-enable-database-replication

```
xe sr-enable-database-replication uuid=sr_uuid
<!--NeedCopy-->
```

Enables XAPI database replication to the specified (shared) SR.

sr-disable-database-replication

```
xe sr-disable-database-replication uuid=sr_uuid
<!--NeedCopy-->
```

Disables XAPI database replication to the specified SR.

sr-forget

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

The XAPI agent forgets about a specified SR on the XenServer host. When the XAPI agent forgets an SR, 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

```
xe sr-introduce name-label=name physical-size=physical_size type=type
    content-type=content_type uuid=sr_uuid
<!--NeedCopy-->
```

Just places an SR record into the database. Use `device-config` to specify additional parameters in the form `device-config:parameter_key=parameter_value`, for example:

```
xe sr-introduce device-config:device=/dev/sdb1
<!--NeedCopy-->
```

> **Note:**
>
> This command is never used in normal operation. This advanced operation might be useful when an SR must be reconfigured as shared after it was created or to help recover from various failure scenarios.

sr-probe

```
xe sr-probe type=type [host-uuid=host_uuid] [device-config:config_name=
    value]
<!--NeedCopy-->
```

Performs a scan of the backend, using the provided `device-config` keys. If the `device-config` is complete for the SR back-end, this command returns a list of the SRs present on the device, if any. If the `device-config` parameters are only partial, a back-end-specific scan is performed, returning results that guide you in improving the remaining `device-config` parameters. The scan results are returned as XML specific to the back end, printed on the CLI.

The exact `device-config` parameters differ depending on the device `type`. For details of these parameters across the different storage back-ends, see Storage.

sr-probe-ext

```
xe sr-probe-ext type=type [host-uuid=host_uuid] [device-config:=config]
     [sm-config:-sm_config]
<!--NeedCopy-->
```

Perform a storage probe. The device-config parameters can be specified by for example device-config:devs=/dev/sdb1. Unlike sr-probe, this command returns results in the same human-readable format for every SR type.

sr-scan

```
xe sr-scan uuid=sr_uuid
<!--NeedCopy-->
```

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

sr-update

```
xe sr-update uuid=uuid
<!--NeedCopy-->
```

Refresh the fields of the SR object in the database.

lvhd-enable-thin-provisioning

```
xe lvhd-enable-thin-provisioning  sr-uuid=sr_uuid initial-allocation=
    initial_allocation allocation-quantum=allocation_quantum
<!--NeedCopy-->
```

Enable thin-provisioning on an LVHD SR.

Subject commands

Commands for working with subjects.

session-subject-identifier-list

```
xe session-subject-identifier-list
<!--NeedCopy-->
```

Return a list of all the user subject ids of all externally-authenticated existing sessions.

session-subject-identifier-logout

```
xe session-subject-identifier-logout subject-identifier=
    subject_identifier
<!--NeedCopy-->
```

Log out all externally-authenticated sessions associated to a user subject id.

session-subject-identifier-logout-all

```
xe session-subject-identifier-logout-all
<!--NeedCopy-->
```

Log out all externally-authenticated sessions.

subject-add

```
xe subject-add subject-name=subject_name
<!--NeedCopy-->
```

Add a subject to the list of subjects that can access the pool.

subject-remove

```
xe subject-remove subject-uuid=subject_uuid
<!--NeedCopy-->
```

Remove a subject from the list of subjects that can access the pool.

subject-role-add

```
xe subject-role-add uuid=uuid [role-name=role_name] [role-uuid=
    role_uuid]
<!--NeedCopy-->
```

Add a role to a subject.

subject-role-remove

```
xe subject-role-remove uuid=uuid [role-name=role_name] [role-uuid=
    role_uuid]
<!--NeedCopy-->
```

Remove a role from a subject.

secret-create

```
xe secret-create value=value
<!--NeedCopy-->
```

Create a secret.

`secret-destroy`

```
xe secret-destroy uuid=uuid
<!--NeedCopy-->
```

Destroy a secret.

Task commands

Commands for working with long-running asynchronous tasks. These commands are tasks such as starting, stopping, and suspending a virtual machine. The tasks 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. For more information, see Low-level parameter commands

Task parameters

Tasks have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the Task	Read only
`name-label`	The name of the Task	Read only
`name-description`	The description string of the Task	Read only
`resident-on`	The unique identifier/object reference of the host on which the task is running	Read only
`status`	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, the value is 1.	Read only

Parameter Name	Description	Type
`type`	If the Task has successfully completed, this parameter contains the type of the encoded result. The type 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 (that is, 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 subtask of	Read only
`subtasks`	Contains the UUIDs of all the subtasks of this task	Read only

task-cancel

```
xe task-cancel [uuid=task_uuid]
<!--NeedCopy-->
```

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 are generic "raw"VMs that can boot an OS vendor installation CD (for example: RHEL, CentOS, SLES, Windows). 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. For more information, see Low-level parameter commands

> **Note:**
>
> Templates cannot be directly converted into VMs by setting the `is-a-template` parameter to **`false`**. Setting `is-a-template` parameter to **`false`** is not supported and results in a VM that cannot be started.

VM template parameters

Templates have the following parameters:

- `uuid` (read only) the unique identifier/object reference for the template
- `name-label` (read/write) the name of the template
- `name-description` (read/write) the description string of the template
- `user-version` (read/write) string for creators of VMs and templates to put version information
- `is-a-template` (read/write) true if this VM is a template.
  Template VMs can never be started, they are used only for cloning other VMs. After this value has been set to true, it cannot be reset to false. Template VMs cannot be converted into VMs using this parameter.

  You can convert a VM to a template with:

  ```
  xe vm-param-set uuid=<vm uuid> is-a-template=true
  <!--NeedCopy-->
  ```

- `is-control-domain` (read only) true if this is a control domain (domain 0 or a driver domain)
- `power-state` (read only) current power state. The value is always halted for a template

- `memory-dynamic-max` (read only) 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`.
- `memory-dynamic-min` (read/write) dynamic minimum memory in bytes.
  Currently unused, but if changed the same constraints for `memory-dynamic-max` must be obeyed.
- `memory-static-max` (read/write) statically set (absolute) maximum memory in bytes. This field is the main value used to determine the amount of memory assigned to a VM.
- `memory-static-min` (read/write) statically set (absolute) minimum memory in bytes.
  This field represents the absolute minimum memory, and `memory-static-min` must be less than `memory-static-max`.
  This value is unused in normal operation, but the previous constraint must be obeyed.
- `suspend-VDI-uuid` (read only) the VDI that a suspend image is stored on (has no meaning for a template)
- `VCPUs-params` (read/write map parameter) configuration parameters for the selected vCPU policy.

  You can tune a vCPU's pinning with:

  ```
  xe template-param-set uuid=<template_uuid> vCPUs-params:mask
      =1,2,3
  <!--NeedCopy-->
  ```

  A VM created from this template 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 template-param-set uuid=<template_uuid> VCPUs-params:weight
      =512 xe template-param-set uuid=<template_uuid> VCPUs-params:
      cap=100
  <!--NeedCopy-->
  ```

  A VM based on this template with a weight of 512 get twice as much CPU as a domain with a weight of 256 on a contended 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 can 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, and so on The default, 0, means that there is no upper cap.
- `VCPUs-max` (read/write) maximum number of vCPUs
- `VCPUs-at-startup` (read/write) boot number of vCPUs

- `actions-after-crash` (read/write) action to take when a VM based on this template crashes
- `console-uuids` (read only set parameter) virtual console devices
- `platform` (read/write map parameter) platform specific configuration

  To disable the emulation of a parallel port for guests:

  ```
  xe vm-param-set uuid=<vm_uuid> platform:parallel=none
  <!--NeedCopy-->
  ```

  To disable the emulation of a serial port:

  ```
  xe vm-param-set uuid=<vm_uuid> platform:hvm_serial=none
  <!--NeedCopy-->
  ```

  To disable the emulation of a USB controller and a USB tablet device:

  ```
  xe vm-param-set uuid=<vm_uuid> platform:usb=false
  xe vm-param-set uuid=<vm_uuid> platform:usb_tablet=false
  <!--NeedCopy-->
  ```

- `allowed-operations` (read only set parameter) list of the operations allowed in this state
- `current-operations` (read only set parameter) list of the operations that are currently in progress on this template
- `allowed-VBD-devices` (read only set parameter) 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).
- `allowed-VIF-devices` (read only set parameter) 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).
- `HVM-boot-policy` (read/write) the boot policy for guests. Either BIOS Order or an empty string.
- `HVM-boot-params` (read/write map parameter) the order key controls the 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.
- `PV-kernel` (read/write) path to the kernel
- `PV-ramdisk` (read/write) path to the `initrd`
- `PV-args` (read/write) string of kernel command line arguments
- `PV-legacy-args` (read/write) string of arguments to make legacy VMs based on this template boot

- `PV-bootloader` (read/write) name of or path to bootloader
- `PV-bootloader-args` (read/write) string of miscellaneous arguments for the bootloader
- `last-boot-CPU-flags` (read only) describes the CPU flags on which a VM based on this template was last booted; not populated for a template
- `resident-on` (read only) the XenServer host on which a VM based on this template is resident. Appears as `not in database` for a template
- `affinity` (read/write) the 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
- `other-config` (read/write map parameter) list of key/value pairs that specify extra configuration parameters for the template
- `start-time` (read only) 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
- `install-time` (read only) 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
- `memory-actual` (read only) the actual memory being used by a VM based on this template; 0 for a template
- `VCPUs-number` (read only) the number of virtual CPUs assigned to a VM based on this template; 0 for a template
- `VCPUs-Utilization` (read only map parameter) 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
- `PV-drivers-version` (read only map parameter) the versions of the paravirtualized drivers for a VM based on this template. Appears as `not in database` for a template
- `PV-drivers-detected` (read only) flag for latest version of the paravirtualized drivers for a VM based on this template. Appears as `not in database` for a template
- `memory` (read only map parameter) memory metrics reported by the agent on a VM based on this template. Appears as `not in database` for a template
- `disks` (read only map parameter) disk metrics reported by the agent on a VM based on this template. Appears as `not in database` for a template
- `networks` (read only map parameter) network metrics reported by the agent on a VM based on this template. Appears as `not in database` for a template

- `other` (read only map parameter) other metrics reported by the agent on a VM based on this template. Appears as `not in database` for a template
- `guest-metrics-last-updated` (read only) timestamp when the in-guest agent performed the last write to these fields. In the form `yyyymmddThh:mm:ss z`, where z is the single-letter military timezone indicator, for example, Z for UTC (GMT)
- `actions-after-shutdown` (read/write) action to take after the VM has shutdown
- `actions-after-reboot` (read/write) action to take after the VM has rebooted
- `possible-hosts` (read only) list of hosts that can potentially host the VM
- `HVM-shadow-multiplier` (read/write) multiplier applied to the amount of shadow that is made available to the guest
- `dom-id` (read only) domain ID (if available, -1 otherwise)
- `recommendations` (read only) XML specification of recommended values and ranges for properties of this VM
- `xenstore-data` (read/write map parameter) data to be inserted into the `xenstore` tree (`/local/domain/*domid*/vmdata`) after the VM is created.
- `is-a-snapshot` (read only) True if this template is a VM snapshot
- `snapshot_of` (read only) the UUID of the VM that this template is a snapshot of
- `snapshots` (read only) the UUIDs of any snapshots that have been taken of this template
- `snapshot_time` (read only) the timestamp of the most recent VM snapshot taken
- `memory-target` (read only) the target amount of memory set for this template
- `blocked-operations` (read/write map parameter) lists the operations that cannot be performed on this template
- `last-boot-record` (read only) record of the last boot parameters for this template, in XML format
- `ha-always-run` (read/write) True if an instance of this template is always restarted on another host if there is a failure of the host it is resident on. This parameter is now deprecated. Use the `ha-restartpriority` parameter instead.
- `ha-restart-priority` (read only) restart or best-effort read/write blobs binary data store
- `live` (read only) relevant only to a running VM.

`template-export`

```
xe template-export template-uuid=uuid_of_existing_template filename=
    filename_for_new_template
<!--NeedCopy-->
```

Exports a copy of a specified template to a file with the specified new file name.

`template-uninstall`

```
xe template-uninstall template-uuid=template_uuid [--force]
<!--NeedCopy-->
```

Uninstall a custom template. This operation will destroy those VDIs that are marked as 'owned' by this template.

Update commands

The following section contains XenServer host update commands.

The update objects can be listed with the standard object listing command (`xe update-list`), and the parameters manipulated with the standard parameter commands. For more information, see Low-level parameter commands

Update parameters

XenServer host updates have the following parameters:

Parameter Name	Description	Type
`uuid`	The unique identifier/object reference for the update	Read only
`host`	The list of hosts that this update is applied to	Read only
`host-uuid`	The unique identifier for the XenServer host to query	Read only
`name-label`	The name of the update	Read only
`name-description`	The description string of the update	Read only
`applied`	Whether or not the update has been applied; true or false	Read only
`installation-size`	The size of the update in bytes	Read only

Parameter Name	Description	Type
`after-apply-guidance`	Whether the XAPI toolstack or the host requires a restart	Read only
`version`	The version of the update	Read only

update-upload

```
xe update-upload file-name=update_filename
<!--NeedCopy-->
```

Upload a specified update file to the XenServer host. This command prepares an update to be applied. On success, the UUID of the uploaded update is printed. If the update has previously been uploaded, `UPDATE_ALREADY_EXISTS` error is returned instead and the patch is not uploaded again.

update-precheck

```
xe update-precheck uuid=update_uuid host-uuid=host_uuid
<!--NeedCopy-->
```

Run the prechecks contained within the specified update on the specified XenServer host.

update-destroy

```
xe update-destroy uuid=update_file_uuid
<!--NeedCopy-->
```

Deletes an update file that has not been applied from the pool. Can be used to delete an update file that cannot be applied to the hosts.

update-apply

```
xe update-apply host-uuid=host_uuid uuid=update_file_uuid
<!--NeedCopy-->
```

Apply the specified update file.

update-pool-apply

```
xe update-pool-apply uuid=update_uuid
<!--NeedCopy-->
```

Apply the specified update to all XenServer hosts in the pool.

`update-introduce`

```
xe update-introduce vdi-uuid=vdi_uuid
<!--NeedCopy-->
```

Introduce update VDI.

`update-pool-clean`

```
xe update-pool-clean uuid=uuid
<!--NeedCopy-->
```

Removes the update's files from all hosts in the pool.

User commands

`user-password-change`

```
xe user-password-change old=old_password new=new_password
<!--NeedCopy-->
```

Changes the password of the logged-in user. The old password field is not checked because you require supervisor privilege to use this command.

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). The VDI has the information on the physical attributes of the virtual disk (which type of SR, whether the disk is sharable, 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. For more information, see Low-level parameter commands

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 VBD represents an empty drive	Read only
`device`	The device seen by the guest, for example `hda`	Read only
`userdevice`	Device number specified by the device parameter during `vbd-create`, for example, 0 for `hda`, 1 for `hdb`, and so on	Read/write
`bootable`	True if this VBD is bootable	Read/write
`mode`	The mode the VBD is 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 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

Parameter Name	Description	Type
qos_algorithm_type	The prioritization algorithm to use	Read/write
qos_algorithm_params	Parameters for the chosen prioritization algorithm	Read/write map parameter
qos_supported_algorithms	Supported prioritization 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 supports hot unplug	Read/write
attachable	True if the device can be attached	Read only
other-config	Extra configuration	Read/write map parameter

vbd-create

```
xe vbd-create vm-uuid=uuid_of_the_vm device=device_value vdi-uuid=
    uuid_of_vdi_to_connect_to [bootable=true] [type=Disk|CD] [mode=RW|RO
    ]
<!--NeedCopy-->
```

Create a VBD on a VM.

The allowable values for the device field are integers 0–15, and the number must be unique for each VM. The current allowable values can be seen in the allowed-VBD-devices parameter on the specified VM. This is seen as userdevice in the vbd parameters.

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 is created for the CD. Mode must be `RO` for a CD.

`vbd-destroy`

```
xe vbd-destroy uuid=uuid_of_vbd
<!--NeedCopy-->
```

Destroy the specified VBD.

If the VBD has its `other-config:owner` parameter set to **`true`**, the associated VDI is also destroyed.

`vbd-eject`

```
xe vbd-eject uuid=uuid_of_vbd
<!--NeedCopy-->
```

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`

```
xe vbd-insert uuid=uuid_of_vbd vdi-uuid=uuid_of_vdi_containing_media
<!--NeedCopy-->
```

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`

```
xe vbd-plug uuid=uuid_of_vbd
<!--NeedCopy-->
```

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

vbd-unplug

```
xe vbd-unplug uuid=uuid_of_vbd
<!--NeedCopy-->
```

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. This is different to the VBD, which is an 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 sharable, whether the media is read/write or read only, and so on). The VBD has the attributes that 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. For more information, see Low-level parameter commands

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

Parameter Name	Description	Type
`virtual-size`	Size of disk as presented to the VM, in bytes. Depending on the storage back-end type, the size may not be respected exactly	Read only
`physical-utilisation`	Amount of physical space that the VDI is 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 when 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`	Extra configuration information for this VDI	Read/write map parameter
`sr-name-label`	Name of the containing storage repository	Read only
`location`	Location information	Read only
`managed`	True if the VDI is managed	Read only
`xenstore-data`	Data to be inserted into the `xenstore` tree (/local/domain/0/backend/vbd/*domid*/*device-id*/smdata) after the VDI is attached. The SM back-ends usually set this field 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

Parameter Name	Description	Type
`snapshots`	The UUIDs 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
`cbt-enabled`	Flag indicating whether changed block tracking is enabled for the VDI	Read/write

`vdi-clone`

```
xe vdi-clone uuid=uuid_of_the_vdi [driver-params:key=value]
<!--NeedCopy-->
```

Create a new, writable copy of the specified VDI that can be used directly. It is a variant of `vdi-copy` that is can expose high-speed image clone facilities where they exist.

Use the optional `driver-params` map parameter to pass extra vendor-specific configuration information to the back-end storage driver that the VDI is based on. For more information, see the storage vendor driver documentation.

`vdi-copy`

```
xe vdi-copy uuid=uuid_of_the_vdi sr-uuid=uuid_of_the_destination_sr
<!--NeedCopy-->
```

Copy a VDI to a specified SR.

`vdi-create`

```
xe vdi-create sr-uuid=uuid_of_sr_to_create_vdi_on name-label=
    name_for_the_vdi type=system|user|suspend|crashdump virtual-size=
    size_of_virtual_disk sm-config-\*=
    storage_specific_configuration_data
<!--NeedCopy-->
```

Create a VDI.

The `virtual-size` parameter can be specified in bytes or using the IEC standard suffixes KiB, MiB, GiB, and TiB.

> **Note:**
>
> SR types that support thin provisioning of disks (such as Local VHD and NFS) do not enforce virtual allocation of disks. Take great care when over-allocating virtual disk space on an SR. If an over-allocated SR becomes full, disk space must be made available either on the SR target substrate or by deleting unused VDIs in the SR.
>
> Some SR types might round up the `virtual-size` value to make it divisible by a configured block size.

`vdi-data-destroy`

```
xe vdi-data-destroy uuid=uuid_of_vdi
<!--NeedCopy-->
```

Destroy the data associated with the specified VDI, but keep the changed block tracking metadata.

> **Note:**
>
> If you use changed block tracking to take incremental backups of the VDI, ensure that you use the `vdi-data-destroy` command to delete snapshots but keep the metadata. Do not use `vdi-destroy` on snapshots of VDIs that have changed block tracking enabled.

`vdi-destroy`

```
xe vdi-destroy uuid=uuid_of_vdi
<!--NeedCopy-->
```

Destroy the specified VDI.

> **Note:**
>
> If you use changed block tracking to take incremental backups of the VDI, ensure that you use the `vdi-data-destroy` command to delete snapshots but keep the metadata. Do not use `vdi-destroy` on snapshots of VDIs that have changed block tracking enabled.
>
> For Local VHD and NFS SR types, disk space is not immediately released on `vdi-destroy`, but periodically during a storage repository scan operation. If you must force deleted disk space to be made available, call `sr-scan` manually.

`vdi-disable-cbt`

```
xe vdi-disable-cbt uuid=uuid_of_vdi
<!--NeedCopy-->
```

Disable changed block tracking for the VDI.

`vdi-enable-cbt`

```
xe vdi-enable-cbt uuid=uuid_of_vdi
<!--NeedCopy-->
```

Enable changed block tracking for the VDI.

> **Note:**
>
> You can enable changed block tracking only on licensed instances of XenServer Premium Edition.

`vdi-export`

```
xe vdi-export uuid=uuid_of_vdi filename=filename_to_export_to [format=
    format] [base=uuid_of_base_vdi] [--progress]
<!--NeedCopy-->
```

Export a VDI to the specified file name. You can export a VDI in one of the following formats:

- `raw`
- `vhd`

The VHD format can be *sparse*. If there are unallocated blocks within the VDI, these blocks might be omitted from the VHD file, therefore making the VHD file smaller. You can export to VHD format from all supported VHD-based storage types (EXT3/EXT4, NFS).

If you specify the `base` parameter, this command exports only those blocks that have changed between the exported VDI and the base VDI.

`vdi-forget`

```
xe vdi-forget uuid=uuid_of_vdi
<!--NeedCopy-->
```

Unconditionally removes a VDI record from the database without touching the storage back-end. In normal operation, use `vdi-destroy` instead.

vdi-import

```
xe vdi-import uuid=uuid_of_vdi filename=filename_to_import_from [format
    =format] [--progress]
<!--NeedCopy-->
```

Import a VDI. You can import a VDI from one of the following formats:

- `raw`
- `vhd`

vdi-introduce

```
xe vdi-introduce uuid=uuid_of_vdi sr-uuid=uuid_of_sr name-label=
    name_of_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]
<!--NeedCopy-->
```

Create a VDI object representing an existing storage device, without actually modifying or creating any storage. This command is primarily used internally to introduce hot-plugged storage devices automatically.

vdi-list-changed-blocks

```
xe vdi-list-changed-blocks vdi-from-uuid=first-vdi-uuid vdi-to-uuid=
    second-vdi-uuid
<!--NeedCopy-->
```

Compare two VDIs and return the list of blocks that have changed between the two as a base64-encoded string. This command works only for VDIs that have changed block tracking enabled.

For more information, see Changed block tracking.

vdi-pool-migrate

```
xe vdi-pool-migrate uuid=VDI_uuid sr-uuid=destination-sr-uuid
<!--NeedCopy-->
```

Migrate a VDI to a specified SR, while the VDI is attached to a running guest. (Storage live migration)

For more information, see Migrate VMs.

vdi-resize

```
xe vdi-resize uuid=vdi_uuid disk-size=new_size_for_disk
<!--NeedCopy-->
```

Change the size of the VDI specified by UUID.

vdi-snapshot

```
xe vdi-snapshot uuid=uuid_of_the_vdi [driver-params=params]
<!--NeedCopy-->
```

Produces a read-write version of a VDI that can be used as a reference for backup or template creation purposes or both. Use the snapshot to perform a backup rather than installing and running backup software inside the VM. The VM continues 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.

Use the optional `driver-params` map parameter to pass extra vendor-specific configuration information to the back-end storage driver that the VDI is based on. For more information, see the storage vendor driver documentation.

A clone of a snapshot always produces a writable VDI.

vdi-unlock

```
xe vdi-unlock uuid=uuid_of_vdi_to_unlock [force=true]
<!--NeedCopy-->
```

Attempts to unlock the specified VDIs. If `force=true` is passed to the command, it forces the unlocking operation.

vdi-update

```
xe vdi-update uuid=uuid
<!--NeedCopy-->
```

Refresh the fields of the VDI object in the database.

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. For more information, see Low-level parameter commands

VIF parameters

VIFs have the following parameters:

- `uuid` (read only) the unique identifier/object reference for the VIF
- `vm-uuid` (read only) the unique identifier/object reference for the VM that this VIF resides on
- `vm-name-label` (read only) the name of the VM that this VIF resides on
- `allowed-operations` (read only set parameter) a list of the operations allowed in this state
- `current-operations` (read only set parameter) a list of the operations that are currently in progress on this VIF
- `device` (read only) integer label of this VIF, indicating the order in which VIF back-ends were created
- `MAC` (read only) MAC address of VIF, as exposed to the VM
- `MTU` (read only) 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
  <!--NeedCopy-->
  ```

- `currently-attached` (read only) true if the device is attached
- `qos_algorithm_type` (read/write) QoS algorithm to use
- `qos_algorithm_params` (read/write map parameter) parameters for the chosen QoS algorithm
- `qos_supported_algorithms` (read only set parameter) supported QoS algorithms for this VIF
- `MAC-autogenerated` (read only) True if the MAC address of the VIF was automatically generated
- `other-config` (read/write map parameter) extra configuration `key:value` pairs
- `other-config:ethtoolrx` (read/write) set to on to enable receive checksum, off to disable

- `other-config:ethtooltx` (read/write) set to on to enable transmit checksum, off to disable
- `other-config:ethtoolsg` (read/write) set to on to enable scatter gather, off to disable
- `other-config:ethtooltso` (read/write) set to on to enable TCP segmentation offload, off to disable
- `other-config:ethtoolufo` (read/write) set to on to enable UDP fragment offload, off to disable
- `other-config:ethtoolgso` (read/write) set to on to enable generic segmentation offload, off to disable
- `other-config:promiscuous` (read/write) 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.
- `network-uuid` (read only) the unique identifier/object reference of the virtual network to which this VIF is connected
- `network-name-label` (read only) the descriptive name of the virtual network to which this VIF is connected
- `io_read_kbs` (read only) average read rate in kB/s for this VIF
- `io_write_kbs` (read only) average write rate in kB/s for this VIF
- `locking_mode` (read/write) Affects the VIFs ability to filter traffic to/from a list of MAC and IP addresses. Requires extra parameters.
- `locking_mode:default` (read/write) Varies according to the default locking mode for the VIF network.

  If the default-locking-mode is set to `disabled`, XenServer applies a filtering rule so that the VIF cannot send or receive traffic. If the default-lockingmode is set to `unlocked`, XenServer removes all the filtering rules associated with the VIF. For more information, see Network Commands.
- `locking_mode:locked` (read/write) Only traffic sent to or sent from the specified MAC and IP addresses is allowed on the VIF. If no IP addresses are specified, no traffic is allowed.
- `locking_mode:unlocked` (read/write) No filters are applied to any traffic going to or from the VIF.
- `locking_mode:disabled` (read/write) XenServer applies a filtering rule is applied so that the VIF drops all traffic.

vif-create

```
xe vif-create vm-uuid=uuid_of_the_vm device=see below network-uuid=
    uuid_of_network_to_connect_to [mac=mac_address]
<!--NeedCopy-->
```

Create a 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 values allowed 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 is created. You can also explicitly set a random MAC address by specifying `mac=random`.

vif-destroy

```
xe vif-destroy uuid=uuid_of_vif
<!--NeedCopy-->
```

Destroy a VIF.

vif-move

```
xe vif-move uuid=uuid network-uuid=network_uuid
<!--NeedCopy-->
```

Move the VIF to another network.

vif-plug

```
xe vif-plug uuid=uuid_of_vif
<!--NeedCopy-->
```

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

vif-unplug

```
xe vif-unplug uuid=uuid_of_vif
<!--NeedCopy-->
```

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

`vif-configure-ipv4`

Configure IPv4 settings for this virtual interface. Set IPv4 settings as below:

```
xe vif-configure-ipv4 uuid=uuid_of_vif mode=static address=CIDR_address
    gateway=gateway_address
<!--NeedCopy-->
```

For example:

```
VIF.configure_ipv4(vifObject,"static", " 192.168.1.10/24", "
   192.168.1.1")
<!--NeedCopy-->
```

Clean IPv4 settings as below:

```
xe vif-configure-ipv4 uuid=uuid_of_vif mode=none
<!--NeedCopy-->
```

`vif-configure-ipv6`

Configure IPv6 settings for this virtual interface. Set IPv6 settings as below:

```
xe vif-configure-ipv6 uuid=uuid_of_vif mode=static address=IP_address
   gateway=gateway_address
<!--NeedCopy-->
```

For example:

```
VIF.configure_ipv6(vifObject,"static", "fd06:7768:b9e5:8b00::5001/64",
   "fd06:7768:b9e5:8b00::1")
<!--NeedCopy-->
```

Clean IPv6 settings as below:

```
xe vif-configure-ipv6 uuid=uuid_of_vif mode=none
<!--NeedCopy-->
```

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. For more information, see PIF commands. For example, to list VLANs, use `xe pif-list`.

`vlan-create`

```
xe vlan-create pif-uuid=uuid_of_pif vlan=vlan_number network-uuid=
    uuid_of_network
<!--NeedCopy-->
```

Create a VLAN on your XenServer host.

`pool-vlan-create`

```
xe pool-vlan-create pif-uuid=uuid_of_pif vlan=vlan_number network-uuid=
    uuid_of_network
<!--NeedCopy-->
```

Create a 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`

```
xe vlan-destroy uuid=uuid_of_pif_mapped_to_vlan
<!--NeedCopy-->
```

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, run `xe vm-list power-state=running`. (Get the full list of fields that can be matched by using the command `xe vm-list params=all`.) For example, specifying `power-state=halted` selects VMs whose `power-state` parameter is equal to `halted`. Where multiple VMs are matching, specify the option `--multiple` to perform the operation. The full list of parameters that can be matched is described at the beginning of this section.

The VM objects can be listed with the standard object listing command (`xe vm-list`), and the parameters manipulated with the standard parameter commands. For more information, see Low-level parameter commands

VM parameters

VMs have the following parameters:

> **Note:**
>
> All writable VM parameter values can be changed while the VM is running, but new parameters are *not* applied dynamically and cannot be applied until the VM is rebooted.

- `appliance` (read/write) the appliance/vApp to which the VM belongs
- `uuid` (read only) the unique identifier/object reference for the VM
- `name-label` (read/write) the name of the VM
- `name-description` (read/write) the description string of the VM
- `order` (read/write) for vApp startup/shutdown and for startup after HA failover. VMs with an order value of 0 (zero) are started first, then VMs with an order value of 1, and so on.
- `version` (read only) the number of times this VM has been recovered. If you want to overwrite a new VM with an older version, call `vm-recover`
- `user-version` (read/write) string for creators of VMs and templates to put version information
- `is-a-template` (read/write) False unless this VM is a template. Template VMs can never be started, they are used only for cloning other VMs After this value has been set to true it cannot be reset to false. Template VMs cannot be converted into VMs using this parameter.

  You can convert a VM to a template with:

  ```
  xe vm-param-set uuid=<vm uuid> is-a-template=true
  <!--NeedCopy-->
  ```

- `is-control-domain` (read only) True if this is a control domain (domain 0 or a driver domain)
- `power-state` (read only) current power state
- `start-delay` (read/write) the delay to wait before a call to start up the VM returns in seconds
- `shutdown-delay` (read/write) the delay to wait before a call to shut down the VM returns in seconds
- `memory-dynamic-max` (read/write) dynamic maximum in bytes
- `memory-dynamic-min` (read/write) dynamic minimum in bytes
- `memory-static-max` (read/write) statically set (absolute) maximum in bytes. If you want to change this value, the VM must be shut down.

- `memory-static-min` (read/write) statically set (absolute) minimum in bytes. If you want to change this value, the VM must be shut down.
- `suspend-VDI-uuid` (read only) the VDI that a suspend image is stored on
- `VCPUs-params` (read/write map parameter) 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
  <!--NeedCopy-->
  ```

  The selected VM then runs 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
  <!--NeedCopy-->
  ```

  A VM with a weight of 512 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, and so on The default, 0, means that there is no upper cap.
- `VCPUs-max` (read/write) maximum number of virtual CPUs.
- `VCPUs-at-startup` (read/write) boot number of virtual CPUs
- `actions-after-crash` (read/write) 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)
  - `destroy` (no coredump and leave VM halted)
- `console-uuids` (read only set parameter) virtual console devices
- `platform` (read/write map parameter) platform-specific configuration

  To disable VDA to switch Windows 10 into Tablet mode:

  ```
  xe vm-param-set uuid=<vm_uuid> platform:acpi_laptop_slate=0
  <!--NeedCopy-->
  ```

To enable VDA to switch Windows 10 into Tablet mode:

```
xe vm-param-set uuid=<vm_uuid> platform:acpi_laptop_slate=1
<!--NeedCopy-->
```

To check current state:

```
xe vm-param-get uuid=<vm_uuid> param-name=platform param-key=
    acpi_laptop_slate
<!--NeedCopy-->
```

- `allowed-operations` (read only set parameter) list of the operations allowed in this state
- `current-operations` (read only set parameter) a list of the operations that are currently in progress on the VM
- `allowed-VBD-devices` (read only set parameter) 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 might not work).
- `allowed-VIF-devices` (read only set parameter) 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 might not work).
- `HVM-boot-policy` (read/write) the boot policy for guests. Either BIOS Order or an empty string.
- `HVM-boot-params` (read/write map parameter) the order key controls the 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.
- `HVM-shadow-multiplier` (read/write) Floating point value which controls the amount of shadow memory overhead to grant the VM. Defaults to 1.0 (the minimum value), and only change this value if you are an advanced user.
- `PV-kernel` (read/write) path to the kernel
- `PV-ramdisk` (read/write) path to the `initrd`
- `PV-args` (read/write) string of kernel command line arguments
- `PV-legacy-args` (read/write) string of arguments to make legacy VMs boot
- `PV-bootloader` (read/write) name of or path to bootloader
- `PV-bootloader-args` (read/write) string of miscellaneous arguments for the bootloader
- `last-boot-CPU-flags` (read only) describes the CPU flags on which the VM was last booted
- `resident-on` (read only) the XenServer host on which a VM is resident

- `affinity` (read/write) The XenServer host which the VM has preference for running on. Used by the `xe vm-start` command to decide where to run the VM
- `other-config` (read/write map parameter) A list of key/value pairs that specify extra configuration parameters for the VM.

  For example, the `other-config` key/value pair `auto_poweron: true` requests to start the VM automatically after any host in the pool boots. You must also set this parameter in your pool's `other-config`. These parameters are now deprecated. Use the `ha-restart-priority` parameter instead.
- `start-time` (read only) timestamp of the date and time that the metrics for the VM were read. This timestamp is in the form `yyyymmddThh:mm:ss z`, where z is the single letter military timezone indicator, for example, Z for UTC (GMT)
- `install-time` (read only) timestamp of the date and time that the metrics for the VM were read. This timestamp is in the form `yyyymmddThh:mm:ss z`, where z is the single letter military timezone indicator, for example, Z for UTC (GMT)
- `memory-actual` (read only) the actual memory being used by a VM
- `VCPUs-number` (read only) the number of virtual CPUs assigned to the VM for a Linux VM. This number can differ from `VCPUS-max` and can be changed without rebooting the VM using the `vm-vcpu-hotplug` command. For more information, see `vm-vcpu-hotplug`. Windows VMs always run with the number of vCPUs set to `VCPUsmax` and must be rebooted to change this value. Performance drops sharply when you set `VCPUs-number` to a value greater than the number of physical CPUs on the XenServer host.
- `VCPUs-Utilization` (read only map parameter) a list of virtual CPUs and their weight
- `os-version` (read only map parameter) the version of the operating system for the VM
- `PV-drivers-version` (read only map parameter) the versions of the paravirtualized drivers for the VM
- `PV-drivers-detected` (read only) flag for latest version of the paravirtualized drivers for the VM
- `memory` (read only map parameter) memory metrics reported by the agent on the VM
- `disks` (read only map parameter) disk metrics reported by the agent on the VM
- `networks` (read only map parameter) network metrics reported by the agent on the VM
- `other` (read only map parameter) other metrics reported by the agent on the VM
- `guest-metrics-lastupdated` (read only) timestamp when the in-guest agent performed the last write to these fields. The timestamp is in the form `yyyymmddThh:mm:ss z`, where z is the single letter military timezone indicator, for example, Z for UTC (GMT)

- `actions-after-shutdown` (read/write) action to take after the VM has shutdown
- `actions-after-reboot` (read/write) action to take after the VM has rebooted
- `possible-hosts` potential hosts of this VM read only
- `dom-id` (read only) domain ID (if available, -1 otherwise)
- `recommendations` (read only) XML specification of recommended values and ranges for properties of this VM
- `xenstore-data` (read/write map parameter) data to be inserted into the `xenstore` tree (`/local/domain/*domid*/vm-data`) after the VM is created
- `is-a-snapshot` (read only) True if this VM is a snapshot
- `snapshot_of` (read only) the UUID of the VM that this snapshot is of
- `snapshots` (read only) the UUIDs of all snapshots of this VM
- `snapshot_time` (read only) the timestamp of the snapshot operation that created this VM snapshot
- `memory-target` (read only) the target amount of memory set for this VM
- `blocked-operations` (read/write map parameter) lists the operations that cannot be performed on this VM
- `last-boot-record` (read only) record of the last boot parameters for this template, in XML format
- `ha-always-run` (read/write) True if this VM is always restarted on another host if there is a failure of the host it is resident on. This parameter is now deprecated. Use the `ha-restart-priority` parameter instead.
- `ha-restart-priority` (read/write) restart or best-effort
- `blobs` (read only) binary data store
- `live` (read only) True if the VM is running. False if HA suspects that the VM is not be running.

vm-assert-can-be-recovered

```
xe vm-assert-can-be-recovered uuid [database] vdi-uuid
<!--NeedCopy-->
```

Tests whether storage is available to recover this VM.

vm-call-plugin

```
xe vm-call-plugin vm-uuid=vm_uuid plugin=plugin fn=function [args:key=
    value]
<!--NeedCopy-->
```

Calls the function within the plug-in on the given VM with optional arguments (args:key=value). To pass a ”value” string with special characters in it (for example new line), an alternative syntax args:key:file=local_file can be used in place, where the content of local_file will be retrieved and assigned to ”key” as a whole.

vm-cd-add

```
xe vm-cd-add cd-name=name_of_new_cd device=
    integer_value_of_an_available_vbd [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Add a new virtual CD to the selected VM. Select the `device` parameter from the value of the `allowed-VBD-devices` parameter of the VM.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-cd-eject

```
xe vm-cd-eject [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Eject a CD from the virtual CD drive. This command only works if exactly one CD is 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 is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-cd-insert

```
xe vm-cd-insert cd-name=name_of_cd [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Insert a CD into the virtual CD drive. This command only works if there is exactly 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 is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-cd-list`

```
xe vm-cd-list [vbd-params] [vdi-params] [vm-selector=vm_selector_value
    ...]
<!--NeedCopy-->
```

Lists CDs attached to the specified VMs.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, 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`

```
xe vm-cd-remove cd-name=name_of_cd [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Remove a virtual CD from the specified VMs.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-checkpoint`

```
xe vm-checkpoint new-name-label=name_label [new-name-description=
    description]
<!--NeedCopy-->
```

Checkpoint an existing VM, using storage-level fast disk snapshot operation where available.

`vm-clone`

```
xe vm-clone new-name-label=name_for_clone [new-name-description=
    description_for_clone] [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

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 is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-compute-maximum-memory`

```
xe 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...]
<!--NeedCopy-->
```

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 later.

For example:

```
xe vm-compute-maximum-memory vm=testvm total=`xe host-list params=
    memory-free --minimal`
<!--NeedCopy-->
```

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 is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-compute-memory-overhead`

```
xe vm-compute-memory-overhead
<!--NeedCopy-->
```

Computes the virtualization memory overhead of a VM.

vm-copy

```
xe vm-copy new-name-label=name_for_copy [new-name-description=
    description_for_copy] [sr-uuid=uuid_of_sr] [vm-selector=
    vm_selector_value...]
<!--NeedCopy-->
```

Copy an existing VM, but without using storage-level fast disk clone operation (even if this option 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 is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-copy-bios-strings

```
xe vm-copy-bios-strings host-uuid=host_uuid
<!--NeedCopy-->
```

Copy the BIOS strings of the given host to the VM.

> **Note:**
>
> After you first start a VM, you cannot change its BIOS strings. Ensure that the BIOS strings are correct before starting the VM for the first time.

vm-crashdump-list

```
xe vm-crashdump-list [vm-selector=vm selector value...]
<!--NeedCopy-->
```

List crashdumps associated with the specified VMs.

When you use the optional argument `params`, 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 is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-data-source-list`

```
xe vm-data-source-list [vm-selector=vm selector value...]
<!--NeedCopy-->
```

List the data sources that can be recorded for a VM.

Select the VMs on which to perform this operation by using the standard selection mechanism. For more information, 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 is performed on all VMs.

Data sources have two parameters –`standard` and `enabled` –which you can seen in the output of this command. If a data source has `enabled` set to **`true`**, the metrics are currently being recorded to the performance database. If a data source has `standard` set to **`true`**, the metrics are recorded to the performance database by default (and `enabled` is also set to **`true`** for this data source). If a data source has `standard` set to **`false`**, the metrics are *not* recorded to the performance database by default (and `enabled` is also 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 command sets `enabled` to **`true`**. To stop, run the `vm-data-source-forget`. This command sets `enabled` to **`false`**.

`vm-data-source-record`

```
xe vm-data-source-record data-source=name_description_of_data-source [
    vm-selector=vm selector value...]
<!--NeedCopy-->
```

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 VMs. For performance reasons, this database is distinct from the normal agent database.

Select the VMs on which to perform this operation by using the standard selection mechanism. For more information, 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 is performed on all VMs.

`vm-data-source-forget`

```
xe vm-data-source-forget data-source=name_description_of_data-source [
    vm-selector=vm selector value...]
<!--NeedCopy-->
```

Stop recording the specified data source for a VM and forget all of the recorded data.

Select the VMs on which to perform this operation by using the standard selection mechanism. For more information, 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 is performed on all VMs.

`vm-data-source-query`

```
xe vm-data-source-query data-source=name_description_of_data-source [vm
    -selector=vm_selector_value...]
<!--NeedCopy-->
```

Display the specified data source for a VM.

Select the VMs on which to perform this operation by using the standard selection mechanism. For more information, 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 is performed on all VMs.

`vm-destroy`

```
xe vm-destroy uuid=uuid_of_vm
<!--NeedCopy-->
```

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`

```
xe vm-disk-add disk-size=size_of_disk_to_add device=uuid_of_device [vm-
    selector=vm_selector_value...]
<!--NeedCopy-->
```

Add a 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, MiB, GiB, and TiB.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-disk-list

```
xe vm-disk-list [vbd-params] [vdi-params] [vm-selector=
    vm_selector_value...]
<!--NeedCopy-->
```

Lists disks attached to the specified VMs. The `vbd-params` and `vdi-params` parameters control the fields of the respective objects to output. Give the parameters as a comma-separated list, or the special key `all` for the complete list.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-disk-remove

```
xe vm-disk-remove device=integer_label_of_disk [vm-selector=
    vm_selector_value...]
<!--NeedCopy-->
```

Remove a disk from the specified VMs and destroy it.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-export

```
xe vm-export filename=export_filename [metadata=true|false] [vm-
    selector=vm_selector_value...]
<!--NeedCopy-->
```

Export the specified VMs (including disk images) to a file on the local machine. Specify the file name to export the VM into using the `filename` parameter. By convention, the file name has a `.xva` extension.

If the `metadata` parameter is **true**, the disks are not exported. Only the VM metadata is written to the output file. Use this parameter when the underlying storage is transferred through other mechanisms, and permits the VM information to be recreated. For more information, see `vm-import`.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-import

```
xe vm-import filename=export_filename [metadata=true|false] [preserve=
    true|false][sr-uuid=destination_sr_uuid]
<!--NeedCopy-->
```

Import a VM from a previously exported file. If `preserve` is set to **true**, the MAC address of the original VM is preserved. The `sr-uuid` determines the destination SR to import the VM into. If this parameter is not specified, the default SR is used.

If the `metadata` is **true**, you can import a previously exported set of metadata without their associated disk blocks. Metadata-only import fails if any VDIs cannot be found (named by SR and `VDI.location`) unless the `--force` option is specified, in which case the import proceeds regardless. If disks can be mirrored or moved out-of-band, metadata import/export is a fast way of moving VMs between disjoint pools. For example, as part of a disaster recovery plan.

> **Note:**
>
> Multiple VM imports are performed faster in serial that in parallel.

vm-install

```
xe vm-install new-name-label=name [template-uuid=
    uuid_of_desired_template] [template=template_uuid_or_name] [sr-uuid=
    sr_uuid | sr-name-label=name_of_sr][copy-bios-strings-from=host_uuid
    ]
<!--NeedCopy-->
```

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 are created in the same SR as these existing disks. Where the SR supports it, these disks are fast copies. If a different SR is specified on the command line, the new disks are created there. In this case, a fast copy is

> not possible and the disks are full copies.
>
> When installing from a template that doesn't have existing disks, any new disks are created in the SR specified, or the pool default SR when an SR is not specified.

`vm-is-bios-customized`

```
xe vm-is-bios-customized
<!--NeedCopy-->
```

Indicates whether the BIOS strings of the VM have been customized.

`vm-memory-dynamic-range-set`

```
xe vm-memory-dynamic-range-set min=min max=max
<!--NeedCopy-->
```

Configure the dynamic memory range of a VM. The dynamic memory range defines soft lower and upper limits for a VM's memory. It's possible to change these fields when a VM is running or halted. The dynamic range must fit within the static range.

`vm-memory-limits-set`

```
xe vm-memory-limits-set static-min=static_min static-max=static_max
    dynamic-min=dynamic_min dynamic-max=dynamic_max
<!--NeedCopy-->
```

Configure the memory limits of a VM.

`vm-memory-set`

```
xe vm-memory-set memory=memory
<!--NeedCopy-->
```

Configure the memory allocation of a VM.

`vm-memory-shadow-multiplier-set`

```
xe vm-memory-shadow-multiplier-set [vm-selector=vm_selector_value...] [
    multiplier=float_memory_multiplier]
<!--NeedCopy-->
```

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 Virtual Apps, 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 host memory decreases according to the multiplier and the `HVM_shadow_multiplier` field is updated with the value that Xen has assigned to the VM. If there is not enough XenServer host memory free, an error is returned.

The VMs on which to perform this operation are selected using the standard selection mechanism. For more information, see VM selectors.

vm-memory-static-range-set

```
xe vm-memory-static-range-set min=min max=max
<!--NeedCopy-->
```

Configure the static memory range of a VM. The static memory range defines hard lower and upper limits for a VM's memory. It's possible to change these fields only when a VM is halted. The static range must encompass the dynamic range.

vm-memory-target-set

```
xe vm-memory-target-set target=target
<!--NeedCopy-->
```

Set the memory target for a halted or running VM. The given value must be within the range defined by the VM's memory_static_min and memory_static_max values.

vm-migrate

```
xe vm-migrate [compress=true|false] [copy=true|false] [host-uuid=
    destination_host_uuid] [host=name_or_ uuid_of_destination_host] [
    force=true|false] [live=true|false] [vm-selector=vm_selector_value
    ...] [remote-master=destination_pool_master_uuid] [remote-username=
    destination_pool_username] [remote-password=
    destination_pool_password] [remote-network=
    destination_pool_network_uuid ][vif:source_vif_uuid=
    destination_network_uuid] [vdi:vdi_uuid=destination_sr_uuid]
<!--NeedCopy-->
```

This command migrates the specified VMs between physical hosts.

The `compress` parameter overrides the `migration-compression` pool parameter for `xe pool-param-set`.

The `host` parameter in the `vm-migrate` command can be either the name or the UUID of the XenServer host. For example, to migrate the VM to another host in the pool, where the VM disks are on storage shared by both hosts:

```
xe vm-migrate uuid=vm_uuid host-uuid=destination_host_uuid
<!--NeedCopy-->
```

To move VMs between hosts in the same pool that do not share storage (storage live migration):

```
xe vm-migrate uuid=vm_uuid host-uuid=destination_host_uuid \
    remote-master=192.0.2.35 remote-username=username remote-password=
        password
<!--NeedCopy-->
```

For storage live migration, you must provide the host name or IP address, user name, and password for the pool coordinator, even when you are migrating within the same pool.

You can choose the SR where each VDI gets stored:

```
xe vm-migrate uuid=vm_uuid remote-master=192.0.2.35 remote-username=
   username remote-password=password host-uuid=destination_host_uuid \
    vdi:vdi_1=destination_sr1_uuid \
    vdi:vdi_2=destination_sr2_uuid \
    vdi:vdi_3=destination_sr3_uuid
<!--NeedCopy-->
```

Additionally, you can choose which network to attach the VM after migration:

```
xe vm-migrate uuid=vm_uuid \
    vdi1:vdi_1_uuid=destination_sr1_uuid \
    vdi2:vdi_2_uuid=destination_sr2_uuid \
    vdi3:vdi_3_uuid=destination_sr3_uuid \
    vif:source_vif_uuid=destination_network_uuid
<!--NeedCopy-->
```

For cross-pool migration:

```
xe vm-migrate uuid=vm_uuid remote-master=192.0.2.35 \
    remote-username=username remote-password=password \
    host-uuid=destination_host_uuid  \
    vif:source_vif_uuid=destination_network_uuid \
    vdi:vdi_uuid=destination_sr_uuid
<!--NeedCopy-->
```

For more information about storage live migration, live migration, and live VDI migration, see Migrate VMs.

> **Note:**
>
> If you are upgrading from an older version of XenServer or Citrix Hypervisor, you might need to shut down and boot all VMs after migrating your VMs, to ensure that new virtualization features are picked up.

By default, the VM is suspended, migrated, and resumed on the other host. The `live` parameter selects live migration. Live migration keeps the VM running while performing the migration, thus minimizing VM downtime. In some circumstances, such as extremely memory-heavy workloads in the VM, live migration falls back into default mode and suspends the VM for a short time before completing the memory transfer.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

vm-pause

```
xe vm-pause
<!--NeedCopy-->
```

Pause a running VM. Note this operation does not free the associated memory (see `vm-suspend`).

vm-reboot

```
xe vm-reboot [vm-selector=vm_selector_value...] [force=true]
<!--NeedCopy-->
```

Reboot the specified VMs.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, 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 reboot. Where the shutdown is akin to pulling the plug on a physical server.

vm-recover

```
xe vm-recover vm-uuid [database] [vdi-uuid] [force]
<!--NeedCopy-->
```

Recovers a VM from the database contained in the supplied VDI.

`vm-reset-powerstate`

```
xe vm-reset-powerstate [vm-selector=vm_selector_value...] {
 force=true }

<!--NeedCopy-->
```

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, 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 coordinator to reset the power-state of the VMs to be `halted`. Essentially, this command forces the lock on the VM and its disks so it can be started next on another pool host. This call *requires* the force flag to be specified, and fails if it is not on the command-line.

`vm-restart-device-models`

```
xe vm-restart-device-models [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Restart the device model for this VM on the host. While the device model is restarting, you can't stop, start, or migrate the VM. The end user of the VM might see a slight pause and resume in their session.

> **Note:**
>
> For the restart device model action to be supported on a Windows VM, the VM must have the XenServer VM Tools for Windows installed.

`vm-resume`

```
xe vm-resume [vm-selector=vm_selector_value...] [force=true|false] [on=
    host_uuid]
<!--NeedCopy-->
```

Resume the specified VMs.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, 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 pool member to start it on. By default the system determines an appropriate host, which might be any of the members of the pool.

vm-retrieve-wlb-recommendations

```
xe vm-retrieve-wlb-recommendations
<!--NeedCopy-->
```

Retrieve the workload balancing recommendations for the selected VM.

vm-shutdown

```
xe vm-shutdown [vm-selector=vm_selector_value...] [force=true|false]
<!--NeedCopy-->
```

Shut down the specified VM.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, 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-snapshot

```
xe vm-snapshot new-name-label=name_label [new-name-description+
    name_description]
<!--NeedCopy-->
```

Snapshot an existing VM, using storage-level fast disk snapshot operation where available.

vm-start

```
xe vm-start [vm-selector=vm_selector_value...] [force=true|false] [on=
    host_uuid] [--multiple]
<!--NeedCopy-->
```

Start the specified VMs.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, 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 pool member to start the VMs on. By default the system determines an appropriate host, which might be any of the members of the pool.

`vm-suspend`

```
xe vm-suspend [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Suspend the specified VM.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-uninstall`

```
xe vm-uninstall [vm-selector=vm_selector_value...] [force=true|false]
<!--NeedCopy-->
```

Uninstall a VM, destroying its disks (those VDIs that are marked RW and connected to this VM only) in addition to its metadata record. To destroy just the VM metadata, use `xe vm-destroy`.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

`vm-unpause`

```
xe vm-unpause
<!--NeedCopy-->
```

Unpause a paused VM.

`vm-vcpu-hotplug`

```
xe vm-vcpu-hotplug new-vcpus=new_total_vcpu_count [vm-selector=
    vm_selector_value...]
<!--NeedCopy-->
```

Dynamically adjust the number of vCPUs available to a running Linux VM. The number of vCPUs is 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.

Use the `new-vcpus` parameter to define the new *total* number of vCPUs that you want to have after running this command. Do not use this parameter to pass the number of vCPUs you want to add. For example, if you have two existing vCPUs in your VM and want to add two more vCPUs, specify `new-vcpus=4`.

The Linux VM or Windows VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. Optional arguments can be any number of the VM parameters listed at the beginning of this section.

> **Note:**
>
> When running Linux VMs without XenServer VM Tools installed, run the following command on the VM as `root` to ensure the newly hot plugged vCPUs are used: `# for i in /sys/devices/system/cpu/cpu[1-9]*/online; do if [ "$(cat $i)"= 0 ]; then echo 1 > $i; fi; done`

`vm-vif-list`

```
xe vm-vif-list [vm-selector=vm_selector_value...]
<!--NeedCopy-->
```

Lists the VIFs from the specified VMs.

The VM or VMs on which this operation is performed are selected using the standard selection mechanism. For more information, see VM selectors. 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.

Scheduled snapshots

Commands for controlling VM scheduled snapshots and their attributes.

The `vmss` objects can be listed with the standard object listing command (`xe vmss-list`), and the parameters manipulated with the standard parameter commands. For more information, see Low-level parameter commands

`vmss-create`

```
xe vmss-create enabled=True/False name-label=name type=type frequency=
    frequency retained-snapshots=value name-description=description
    schedule:schedule
<!--NeedCopy-->
```

Creates a snapshot schedule in the pool.

For example:

```
xe vmss-create retained-snapshots=9 enabled=true frequency=daily \
    name-description=sample name-label=samplepolicy type=snapshot \
    schedule:hour=10 schedule:min=30
```

```
<!--NeedCopy-->
```

Snapshot schedules have the following parameters:

Parameter Name	Description	Type
`name-label`	Name of the snapshot schedule.	Read/write
`name-description`	Description of the snapshot schedule.	Read/write
`type`	Disk snapshot or memory snapshot.	Read/write
`frequency`	Hourly; Daily; Weekly	Read/write
`retained-snapshots`	Snapshots to be retained. Range: 1-10.	Read/write
`schedule`	`schedule:days` (Monday to Sunday), `schedule:hours` (0 to 23), `schedule:minutes` (0, 15, 30, 45)	Read/write

`vmss-destroy`

```
xe vmss-destroy uuid=uuid
<!--NeedCopy-->
```

Destroys a snapshot schedule in the pool.

USB pass-through

USB pass-through is supported for the following USB versions: 1.1, 2.0, and 3.0.

USB pass-through enable/disable

```
xe pusb-param-set uuid=pusb_uuid passthrough-enabled=true/false
<!--NeedCopy-->
```

Enable/disable USB Pass-through.

`pusb-scan`

```
xe pusb-scan host-uuid=host_uuid
<!--NeedCopy-->
```

Scan PUSB and update.

`vusb-create`

```
xe vusb-create usb-group-uuid=usb_group_uuid vm-uuid=vm_uuid
<!--NeedCopy-->
```

Creates a virtual USB in the pool. Start the VM to pass through the USB to the VM.

`vusb-unplug`

```
xe vusb-unplug uuid=vusb_uuid
<!--NeedCopy-->
```

Unplugs USB from VM.

`vusb-destroy`

```
xe vusb-destroy uuid=vusb_uuid
<!--NeedCopy-->
```

Removes the virtual USB list from VM.

Troubleshooting

April 8, 2024

If you experience technical difficulties with the XenServer host, this section is meant to help you solve the problem if possible. If it isn't possible, use the information in this section to gather the application logs and other data that can help Technical Support track and resolve the issue.

The following articles provide troubleshooting information about specific areas of the product:

- VM troubleshooting
- Networking troubleshooting
- Clustered pool troubleshooting

- XenCenter troubleshooting
- Workload Balancing troubleshooting
- Conversion Manager troubleshooting

Troubleshoot 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 that you are attempting to connect to?

  XenCenter 8.2.7 and earlier are not supported with XenServer 8 hosts. To manage your XenServer 8 hosts or pools, you require the latest version of XenCenter with a version of the form YYYY.x.x.

  To correct this issue, install the latest version of XenCenter.

- Is your license current?

  You can see the expiration date for your license access code in the XenServer host **General** tab under the **License Details** section in XenCenter.

  For more information on licensing a host, see Licensing.

- The XenServer host talks to XenCenter using HTTPS over the following ports:

  - Port 443 (a two-way connection for commands and responses using the management API)
  - Port 5900 for graphical VNC connections with paravirtualized Linux VMs.

  If you have a firewall enabled between the XenServer host and the machine running the client software, ensure that it allows traffic from these ports. For more information, see Internet connectivity.

Gather XenServer and XenCenter logs

XenServer host logs

XenCenter can be used to gather XenServer host information.

Click **Server Status Report** in the **Tools** menu to open the **Server Status Report** task. You can select from a list of different types of information (various logs, crash dumps, and so on). The information is compiled and downloaded to the machine that XenCenter is running on. For more information, see the XenCenter documentation.

By default, the files gathered for a server status report can be limited in size. If you need log files that are larger than the default, you can run the command `xenserver-status-report -u` in the XenServer host console.

> **Important:**
>
> XenServer host logs may contain sensitive information.

Sending host log messages to a central server Rather than have logs written to the control domain filesystem, you can configure your 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.

Set the syslog_destination parameter to the host name or IP address of the remote server where you want the logs to be written:

```
xe host-param-set uuid=host_uuid logging:syslog_destination=hostname
<!--NeedCopy-->
```

Run the command:

```
xe host-syslog-reconfigure uuid=host_uuid
<!--NeedCopy-->
```

To enforce the change. (You can also run this command remotely by specifying the `host` parameter.)

XenCenter logs

XenCenter also has a 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 at the following path: `%userprofile%\AppData\Roaming\XenServer\XenCenter\logs\XenCenter.log`.

To locate the XenCenter log files - for example, when you want to open or email the log file - click **View XenCenter Log Files** in the XenCenter **Help** menu.

Installation logs

If you experience an unknown error during installation, capture the log file from your host and provide it to Technical Support.

Using a keyboard connected directly to the host machine (not connected over a serial port), you can access three virtual terminals during installation:

- Press **Alt+F1** to access the main XenServer Installer

- Press **Alt+F2** to access a local shell
- Press **Alt+F3** to access the event log

To capture and save the log files:

1. Press **Alt+F2** to access the local shell.
2. Enter the following:

   ```
   /opt/xensource/installer/report.py
   <!--NeedCopy-->
   ```

3. You are prompted to choose where you want to save the log file: **NFS**, **FTP**, or **Local media**.

   Select **NFS** or **FTP** to copy the log file to another machine on your network. To do so, networking must be working properly, and you must have write access to a remote machine.

   Select **Local media** to save the file to a removable storage device, such as a USB flash drive, on the local machine.

   Once you have made your selections, the program writes the log file to your chosen location. The file name is `support.tar.bz2`.

Support

April 8, 2024

We provide Technical Support services to customers with a XenServer Premium Edition or Standard Edition license. To access this support, you can open a Support Case online or contact the support center by phone if you experience technical difficulties. For more information, see the XenServer support page.

If you are using XenServer Trial Edition (unlicensed), you cannot access this support, but we do value your feedback. For more information, see Provide feedback for XenServer and XenCenter.

> **Note:**
>
> If you installed XenServer 8 during its preview period, you must apply updates published on March 18, 2024 or later to get your pool to a production-supported level that is eligible to receive technical support.

Frequent updates

XenServer 8 uses a frequent update model that delivers features, fixes, and improvements to your hosts. We expect you to consume these updates within six months to remain in support. If the update level of your pool is older than six months, we will ask you to reproduce the issue on the latest update level.

Support checklist

This section guides you through possible actions to take when you experience an issue in your XenServer environment. By completing as many of these steps as you can, you help us troubleshoot your issue faster.

First steps

When first experiencing the issue, complete these steps:

1. Before carrying out any recovery steps, capture any logs that you can from the environment:

    - If the issue relates to XenServer, capture a server status report (SSR) of the host or pool where the issue was seen. Go to the XenCenter **Tools** menu then **Server Status Report**.
    - If the issue relates to XenCenter:
        - Get the application log files by going to the XenCenter **Help** menu then **View XenCenter Log Files**.
        - Capture screenshots of the relevant displays.
    - If the issue relates to a VM, gather any relevant logs from the VM operating system.

    For more information about getting log information, see Gather XenServer and XenCenter logs.

2. Make a note of the synchronization date, synchronization checksum, and update checksum for your host or pool. For more information, see View pending tasks.

3. If necessary, attempt to recover your environment to a working state.

Self-help

We provide information and guidance that can help you diagnose and resolve issues you might experience.

1. Check this documentation for help:

    - Known Issues: This article lists known issues in XenServer and, if applicable, any workarounds you can apply.
    - Early Access and Normal: These articles list available updates to XenServer. A fix for your issue might have recently been released.

    - Troubleshooting: This article is an entrypoint into the troubleshooting information provided in the documentation.
    - Review the section of the documentation related to the feature that is showing the issue. There might be constraints to this feature that are causing the issue or configuration options that can help fix it.

2. The Citrix Knowledge Center contains many articles written by our Technical Support team that describe solutions to previously seen issues with XenServer.

If you take any diagnostic action or change your environment configuration as a result of this information, make a note of it and let us know if you do contact support.

Update to the latest version

If you are already using the latest version of all your XenServer components, skip to Capture logs.

If you are not running the latest XenServer, XenCenter, or related component, the fix for your issue might be included in the latest version or update level. We recommend that you update your environment to the latest version, if possible.

1. Update your pool to the latest update.
2. Update XenCenter to the latest version.
3. If the issue relates to a Windows VM, update the XenServer VM Tools for Windows to the latest version.
4. If the issue relates to a Linux VM, ensure the latest version of the XenServer VM Tools for Linux is installed. These tools are available on https://xenserver.com/downloads.
5. If the issue relates to Workload Balancing or the XenServer Conversion Manager, ensure you are using the latest version available on https://xenserver.com/downloads.

Reproduce the issue on the latest version

If you updated any of the components in your environment since you encountered the issue, attempt to reproduce the issue now.

Capture logs

- If the issue relates to XenServer, capture a server status report (SSR) of the host or pool where the issue was encountered. Go to the XenCenter **Tools** menu then **Server Status Report**.
- If the issue relates to XenCenter:
    - Get the application log files by going to the XenCenter **Help** menu then **View XenCenter Log Files**.

    - Capture screenshots of the relevant displays.

- If the issue relates to a VM, gather any relevant logs from the VM operating system.

Contact support

For methods of getting in touch with us, see the XenServer support page.

Provide the following information to support:

- Date and time when the issue was encountered
- Pool update channel (Early Access or Normal)
- Pool coordinator's name
- Date and time when the pool was last synchronized
- Synchronization checksum for the pool coordinator
- Date and time when the pool was last updated
- Update checksum for all hosts in the pool
- Any relevant screenshots to help to describe the issue
- Any change or event that might have triggered the issue
- Any known workaround
- Any diagnostic steps that you've already taken
- The SSR for the problem pool
- If applicable, the XenCenter logs

> **Note:**
>
> If you are unable to reproduce the issue on an up-to-date pool, attach the SSR that you captured when the issue first happened and explain why you were unable to apply all updates.

Provide feedback for XenServer and XenCenter

Help us to improve our product by providing feedback on the features and usability of our new release. To provide feedback, do not contact Technical Support but instead submit a feedback email. Trial Edition users can report a bug through the bugs portal.

Submit a feedback email

Email any feedback and queries to feedback@xenserver.com. To help us understand the full context of your situation, ensure that you include the following information in your feedback email:

- Your full name
- Your company or business
- Your geographical location
- Your license type
- The number of hosts in your production deployment
- The guest OS where the issue occurred (Windows or Linux)

Do not use this email address to request Technical Support.

Report a bug (Trial Edition users)

> **Note:**
>
> If you are a Premium Edition or Standard Edition customer, don't use the bugs portal to request assistance. Your issue is addressed faster by getting in touch with Technical Support. For more information, see the XenServer support page.

To report a bug, submit a ticket by using the XenServer 8 Bugs portal.

- Create an account on the XenServer 8 Bugs portal. When creating your account, ensure that you use a valid contactable email. Although we aim to respond to you on the ticket, sometimes it might be necessary for us to contact you directly by email.
- Log a bug on the XenServer 8 Bugs portal by clicking **Report an issue on XenServer**.

Issues raised through the XenServer 8 Bugs portal are triaged and you might be contacted if the issue merits investigation.

Third party notices

May 10, 2023

This release of XenServer includes third-party software licensed under a number of different licenses.

To extract the licensing information from your installed XenServer product and components, see the instructions in XenServer Open Source Licensing and Attribution.

In addition, note the following information:

- This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)
- This product includes cryptographic software written by Eric Young (eay@cryptsoft.com).
- XenServer High Availability is powered by everRun, a registered trademark of Stratus Technologies Bermuda, Limited.

XenServer Open Source Licensing and Attribution

April 8, 2024

The XenServer product is a compilation of software packages. Each package is governed by its own license. The complete licensing terms applicable to a given package can be found in the source RPM of the package, unless the package is covered by a proprietary license which does not permit source redistribution, in which case no source RPM is made available.

The XenServer distribution contains content from CentOS Linux and CentOS Stream. Where the CentOS Project holds any copyright in the packages making up the CentOS Linux or CentOS Stream distributions, that copyright is licensed under the GPLv2 license unless otherwise noted. For more information, see https://www.centos.org/legal/licensing-policy/.

Extracting attribution and licensing information on an installed XenServer host

This article provides a method to extract the licensing information from all RPM packages included in your XenServer installation.

Get overview information

To list all RPMs and their licenses:

1. Connect to your XenServer host console by SSH or through XenCenter.
2. At the console command line, run the following command:

```
rpm -qa --qf '%{
 name }
 -%{
 version }
 : %{
 license }
 \n'
```

This command lists all installed components and the licenses they are distributed under. The output is of the following form:

```
readline-6.2: GPLv3+
gnupg2-2.0.22: GPLv3+
libdb-5.3.21: BSD and LGPLv2 and Sleepycat
rpm-python-4.11.3: GPLv2+
sqlite-3.7.17: Public Domain
qrencode-libs-3.4.1: LGPLv2+
libselinux-2.5: Public Domain
ustr-1.0.4: MIT or LGPLv2+ or BSD
gdbm-1.10: GPLv3+
procps-ng-3.3.10: GPL+ and GPLv2 and GPLv2+ and GPLv3+ and LGPLv2+
p11-kit-trust-0.23.5: BSD
device-mapper-libs-1.02.149: LGPLv2
xenserver-release-8.2.50: GPLv2
elfutils-libs-0.170: GPLv2+ or LGPLv3+
xz-libs-5.2.2: LGPLv2+
dbus-1.10.24: (GPLv2+ or AFL) and GPLv2+
elfutils-libelf-0.170: GPLv2+ or LGPLv3+
systemd-sysv-219: LGPLv2+
jemalloc-3.6.0: BSD
<!--NeedCopy-->
```

Get detailed information

To obtain a more complete list of information about each installed component:

1. Connect to your XenServer host console by SSH or through XenCenter.
2. At the console command line, run the following command:

```
rpm -qai | sed '/^Name /i\\n'
```

The output is of the following form:

```
Name: host-upgrade-plugin
Version     : 2.2.6
Release     : 1.xs8
Architecture: noarch
Install Date: Wed 23 Aug 2023 01:54:25 PM UTC
Group: Unspecified
Size: 101626
License     : GPL
Signature   : RSA/SHA256, Tue 30 May 2023 10:01:44 AM UTC, Key ID
    5259d0b0f6529a4e
Source RPM  : host-upgrade-plugin-2.2.6-1.xs8.src.rpm
Build Date  : Fri 26 May 2023 03:05:49 AM UTC
Build Host  : cf27e1dd25c54cbb8cef79726ed2bf2c
Relocations : (not relocatable)
Packager    : Koji
```

```
Vendor      : Cloud Software Group, Inc.
Summary     : Host upgrade plugin
Description :
Host upgrade plugin.

Name        : m4
Version     : 1.4.16
Release     : 10.el7
Architecture: x86_64
Install Date: Wed 23 Aug 2023 01:52:31 PM UTC
Group       : Applications/Text
Size        : 525707
License     : GPLv3+
Signature   : RSA/SHA256, Tue 09 May 2023 02:53:25 PM UTC, Key ID
    5259d0b0f6529a4e
Source RPM  : m4-1.4.16-10.el7.src.rpm
Build Date  : Fri 20 Nov 2015 07:28:07 AM UTC
Build Host  : worker1.bsys.centos.org
Relocations : (not relocatable)
Packager    : CentOS BuildSystem <http://bugs.centos.org>
Vendor      : CentOS
URL         : http://www.gnu.org/software/m4/
Summary     : The GNU macro processor
Description :
A GNU implementation of the traditional UNIX macro processor.  M4
    is
useful for writing text files which can be logically parsed, and
    is used
by many programs as part of their build process.  M4 has built-in
functions for including files, running shell commands, doing
    arithmetic,
etc.  The autoconf program needs m4 for generating configure
    scripts, but
not for running configure scripts.
<!--NeedCopy-->
```

Multiple licenses Some components in the XenServer product contain multiple licenses. For example, `procps-ng-3.3.10`contains the following parts:

- some parts which are licensed with the original GPL (or any later version)
- some parts which are licensed with the GPL version 2 (only)
- some parts which are licensed with the GPL version 2 (or any later version)
- some parts which are licensed with the GPL version 3 (or any later version)
- some parts which are licensed with the LGPL version 2 (or any later version)

In this case, inspect the documentation in `/usr/share/doc/procps-ng-3.3.10` for further information or, if necessary, the corresponding source RPM.

Get more information

In most cases, further information about each component and full license text is installed in either `/usr/share/doc/` or `/usr/share/licenses`.

For example, you can find more information about the component `jemalloc-3.6.0` by running the following command:

```
ls -l /usr/share/doc/jemalloc-3.6.0/

total 120
-rw-r--r--. 1 root root   1703 Mar 31  2014 COPYING
-rw-r--r--. 1 root root 109739 Mar 31  2014 jemalloc.html
-rw-r--r--. 1 root root   1084 Mar 31  2014 README
-rw-r--r--. 1 root root     50 Mar 31  2014 VERSION
```

However, for some components distributed by CentOS, the license text is not installed in the XenServer product. To view the license text for these components, you can look inside the source RPMs.

Download the source RPMS

We make the source RPMs for the XenServer host available in the following locations:

- For the base ISOs that are periodically released, source files are provided on the XenServer download page.
- For updates, the source files are uploaded to a CDN alongside the product RPMs. To download the source files, complete the following steps:

    1. Log in to the console of the pool coordinator host.
    2. Get the UUIDs of the repositories in use by the pool, by running the following command:

        ```
        xe pool-param-get param-name=repositories uuid=<POOL_UUID>
        ```

    3. Check if a proxy is configured and get its URL, by running the following command:

        ```
        xe pool-param-get param-name=repository-proxy-url uuid=<
            POOL_UUID>
        ```

        You cannot use these steps to download the source files through a proxy that has a user name and password configured.

    4. If a proxy is configured, in yum configure the proxy for the repositories, by running the following command for each repository:

        ```
        yum-config-manager --save --setopt=remote-<REPO_UUID>-source.
            proxy=<PROXY_URL>
        ```

5. To download the source RPM of a package, run the following command:

```
yumdownloader --disablerepo=* --enablerepo=remote-<REPO_UUID_0
    >-source,remote-<REPO_UUID_1>-source --source <PKG_NAME>
```

The name of the source file for a specific component is given by the value of "Source RPM"in the detailed information output. For example:

```
Source RPM : m4-1.4.16-10.el7.src.rpm
<!--NeedCopy-->
```

Other XenServer components

Supplemental Packs

Supplemental packs are installed into the XenServer host. If you have supplemental packs installed in your host, their RPM information is included when you complete the steps in the previous section of this article.

The source files for supplemental packs are also provided on the XenServer download page.

XenCenter

To view information about third-party components included in XenCenter, complete the following steps:

1. In XenCenter, go to **Help > About XenCenter**.
2. Click **View Legal Notices**.

XenServer VM Tools for Windows

The XenServer VM Tools for Windows comprises the following components:

- The Management Agent, which is covered by a proprietary license.
- The Windows I/O drivers, which are covered by the BSD2 license. Copyright Cloud Software Group, Inc.

  Licensing information is included in the INF file for each driver. When the drivers are installed on your Windows system by Windows Update or the management agent installer, the INF files are stored as `C:\Windows\INF\OEM*.inf`. The management agent installer also places the INF files in `C:\Program Files\XenServer\XenTools\Drivers\***.inf`.

Source is not provided for XenServer VM Tools for Windows.

XenServer VM Tools for Linux

The XenServer VM Tools for Linux are covered by the BSD2 license. Copyright Cloud Software Group, Inc.

The archive file provided on the product download page contains the license file and source files for the tools.

Virtual Appliances

The following virtual appliances are provided as optional components for your XenServer environment:

- XenServer Conversion Manager Virtual Appliance
- Workload Balancing Virtual Appliance

These virtual appliances are also CentOS based. You can use the same commands as those given for the XenServer host to get overview and detailed information about the open source packages included in the virtual appliances.

In the console of the virtual appliance, run the following commands:

- For overview information: `rpm -qa --qf '%{ name } -%{ version } : %{ license } \n'`
- For detailed information: `rpm -qai | sed '/^Name /i\\n'`

In addition, the XenServer Conversion Manager virtual appliance and Workload Balancing virtual appliance dynamically use some third-party components.

- For XenServer Conversion Manager virtual appliance, the license files for these components are located at the following path: `/opt/vpxxcm/conversion`.
- For Workload Balancing virtual appliance, the license files for these components are located at the following path: `/opt/vpx/wlb`.

Source files for the virtual appliances are provided on the XenServer download page.

Developer documentation

May 10, 2023

The following developer documentation is available:

- Management API Guide

- Software Development Kit Guide
- Changed Block Tracking Guide
- Supplemental Packs and the DDK Guide
- XenCenter Plug-in Specification Guide

XenServer 8 Management API

January 23, 2024

This document defines the XenServer Management API - an interface for remotely configuring and controlling virtualised guests running on a Xen-enabled host.

The API is presented here as a set of Remote Procedure Calls (RPCs). There are two supported wire formats, one based upon XML-RPC and one based upon JSON-RPC (v1.0 and v2.0 are both recognised). No specific language bindings are prescribed, although examples are given in the Python programming language.

Although we adopt some terminology from object-oriented programming, future client language bindings may or may not be object oriented. The API reference uses the terminology *classes* and *objects*. For our purposes a *class* is simply a hierarchical namespace; an *object* is an instance of a class with its fields set to specific values. Objects are persistent and exist on the server-side. Clients may obtain opaque references to these server-side objects and then access their fields via get/set RPCs.

For each class we specify a list of fields along with their *types* and *qualifiers*. A qualifier is one of:

- `RO`/`runtime`: the field is Read Only. Furthermore, its value is automatically computed at runtime. For example, current CPU load and disk IO throughput.
- `RO`/`constructor`: the field must be manually set when a new object is created, but is then Read Only for the duration of the object's life. For example, the maximum memory addressable by a guest is set before the guest boots.
- `RW`: the field is Read/Write. For example, the name of a VM.

Types

The following types are used to specify methods and fields in the API Reference:

- `string`: Text strings.
- **`int`**: 64-bit integers.
- **`float`**: IEEE double-precision floating-point numbers.
- `bool`: Boolean.
- `datetime`: Date and timestamp.
- `c ref`: Reference to an object of class `c`.
- `t set`: Arbitrary-length set of values of type `t`.
- `(k -> v)map`: Mapping from values of type `k` to values of type `v`.
- `e enum`: Enumeration type with name `e`. Enums are defined in the API reference together with classes that use them.

Note that there are a number of cases where `ref`s are *doubly linked*.
For example, a `VM` has a field called `VIFs` of type `VIF ref set`;
this field lists the network interfaces attached to a particular VM.
Similarly, the `VIF` class has a field called `VM` of type `VM ref`
which references the VM to which the interface is connected.
These two fields are *bound together*, in the sense that
creating a new VIF causes the `VIFs` field of the corresponding
VM object to be updated automatically.

The API reference lists explicitly the fields that are
bound together in this way. It also contains a diagram that shows
relationships between classes. In this diagram an edge signifies the
existence of a pair of fields that are bound together, using standard
crows-foot notation to signify the type of relationship (e.g.
one-many, many-many).

RPCs associated with fields

Each field, `f`, has an RPC accessor associated with it that returns `f`'s value:

- `get_f (r)`: takes a `ref`, `r` that refers to an object and returns the value of `f`.

Each field, `f`, with qualifier `RW` and whose outermost type is `set` has the
following additional RPCs associated with it:

- `add_f(r, v)`: adds a new element `v` to the set.

Note that sets cannot contain duplicate values, hence this operation has no action in the case that `v` is already in the set.

- `remove_f(r, v)`: removes element `v` from the set.

Each field, `f`, with qualifier `RW` and whose outermost type is `map` has the following additional RPCs associated with it:

- `add_to_f(r, k, v)`: adds new pair `k -> v` to the mapping stored in `f` in object`r`. Attempting to add a new pair for duplicate key, `k`, fails with a `MAP_DUPLICATE_KEY` error.
- `remove_from_f(r, k)`: removes the pair with key `k` from the mapping stored in `f` in object `r`.

Each field whose outermost type is neither `set` nor `map`, but whose qualifier is `RW` has an RPC accessor associated with it that sets its value:

- `set_f(r, v)`: sets the field `f` on object `r` to value `v`.

RPCs associated with classes

- Most classes have a *constructor* RPC named `create` that takes as parameters all fields marked `RW` and `RO`/`constructor`. The result of this RPC is that a new *persistent* object is created on the server-side with the specified field values.
- Each class has a `get_by_uuid(uuid)` RPC that returns the object of that class that has the specified `uuid`.
- Each class that has a `name_label` field has a `get_by_name_label(name_label)` RPC that returns a set of objects of that class that have the specified `name_label`.
- Most classes have a `destroy(r)` RPC that explicitly deletes the persistent object specified by `r` from the system. This is a non-cascading delete - if the object being removed is referenced by another object then the `destroy` call will fail.

Apart from the RPCs enumerated above, some classes have additional RPCs associated with them. For example, the `VM` class has RPCs for cloning, suspending, starting etc. Such additional RPCs are described explicitly in the API reference.

Wire Protocol for Remote API Calls

January 23, 2024

API calls are sent over a network to a Xen-enabled host using an RPC protocol.
Here we describe how the higher-level types used in our API Reference are mapped
to primitive RPC types, covering the two supported wire formats
XML-RPC and JSON-RPC.

XML-RPC Protocol

We specify the signatures of API functions in the following style:

```
(VM ref set)  VM.get_all()
<!--NeedCopy-->
```

This specifies that the function with name `VM.get_all` takes
no parameters and returns a `set` of `VM ref`.
These types are mapped onto XML-RPC types in a straight-forward manner:

- the types **`float`**, `bool`, `datetime`, and `string` map directly to the XML-RPC <**`double`**>, <**`boolean`**>, <`dateTime.iso8601`>, and <`string`> elements.
- all `ref` types are opaque references, encoded as the XML-RPC's <`string`> type. Users of the API should not make assumptions about the concrete form of these strings and should not expect them to remain valid after the client's session with the server has terminated.
- fields named `uuid` of type `string` are mapped to the XML-RPC <`string`> type. The string itself is the OSF DCE UUID presentation format (as output by `uuidgen`).
- **`int`** is assumed to be 64-bit in our API and is encoded as a string of decimal digits (rather than using XML-RPC's built-in 32-bit <`i4`> type).
- values of `enum` types are encoded as strings. For example, the value `destroy` of `enum on_normal_exit`, would be conveyed as:

```
    <value><string>destroy</string></value>
<!--NeedCopy-->
```

- for all our types, `t`, our type `t set` simply maps to XML-RPC's <`array`> type, so, for example, a value of type `string set` would be transmitted like this:

```
    <array>
      <data>
        <value><string>CX8</string></value>
        <value><string>PSE36</string></value>
        <value><string>FPU</string></value>
      </data>
    </array>
<!--NeedCopy-->
```

- for types `k` and `v`, our type `(k -> v)map` maps onto an XML-RPC `<struct>`, with the key as the name of the struct. Note that the `(k -> v)map` type is only valid when `k` is a `string`, `ref`, or `int`, and in each case the keys of the maps are stringified as above. For example, the `(string -> float)map` containing the mappings *Mike -> 2.3* and *John -> 1.2* would be represented as:

```
    <value>
      <struct>
        <member>
          <name>Mike</name>
          <value><double>2.3</double></value>
        </member>
        <member>
          <name>John</name>
          <value><double>1.2</double></value>
        </member>
      </struct>
    </value>
<!--NeedCopy-->
```

- our `void` type is transmitted as an empty string.

XML-RPC Return Values and Status Codes

The return value of an RPC call is an XML-RPC `<struct>`.

- The first element of the struct is named `Status`; it contains a string value indicating whether the result of the call was a `Success` or a `Failure`.

If the `Status` is `Success` then the struct contains a second element named `Value`:

- The element of the struct named `Value` contains the function's return value.

If the `Status` is `Failure` then the struct contains a second element named `ErrorDescription`:

- The element of the struct named `ErrorDescription` contains an array of string values. The first element of the array is an error code; the rest of the elements are strings representing error parameters relating to that code.

For example, an XML-RPC return value from the `host.get_resident_VMs` function may look like this:

```
    <struct>
       <member>
         <name>Status</name>
         <value>Success</value>
       </member>
       <member>
          <name>Value</name>
          <value>
            <array>
               <data>
                 <value>81547a35-205c-a551-c577-00b982c5fe00</value>
                 <value>61c85a22-05da-b8a2-2e55-06b0847da503</value>
                 <value>1d401ec4-3c17-35a6-fc79-cee6bd9811fe</value>
               </data>
            </array>
         </value>
       </member>
    </struct>
<!--NeedCopy-->
```

JSON-RPC Protocol

We specify the signatures of API functions in the following style:

```
(VM ref set)  VM.get_all()
<!--NeedCopy-->
```

This specifies that the function with name `VM.get_all` takes no parameters and returns a `set` of `VM ref`. These types are mapped onto JSON-RPC types in the following manner:

- the types **`float`** and `bool` map directly to the JSON types `number` and **`boolean`**, while `datetime` and `string` are represented as the JSON `string` type.
- all `ref` types are opaque references, encoded as the JSON `string` type. Users of the API should not make assumptions about the concrete form of these strings and should not expect them to remain valid after the client's session with the server has terminated.

- fields named `uuid` of type `string` are mapped to the JSON `string` type. The string itself is the OSF DCE UUID presentation format (as output by `uuidgen`).
- **`int`** is assumed to be 64-bit in our API and is encoded as a JSON `number` without decimal point or exponent, preserved as a string.
- values of `enum` types are encoded as the JSON `string` type. For example, the value `destroy` of `enum on_normal_exit`, would be conveyed as:

```
  "destroy"
<!--NeedCopy-->
```

- for all our types, `t`, our type `t set` simply maps to the JSON `array` type, so, for example, a value of type `string set` would be transmitted like this:

```
  [ "CX8", "PSE36", "FPU" ]
<!--NeedCopy-->
```

- for types `k` and `v`, our type `(k -> v) map` maps onto a JSON object which contains members with name `k` and value `v`. Note that the `(k -> v) map` type is only valid when `k` is a `string`, `ref`, or **`int`**, and in each case the keys of the maps are stringified as above. For example, the `(string -> `**`float`**`) map` containing the mappings *Mike -> 2.3* and *John -> 1.2* would be represented as:

```
  {

    "Mike": 2.3,
    "John": 1.2
   }

<!--NeedCopy-->
```

- our **`void`** type is transmitted as an empty string.

Both versions 1.0 and 2.0 of the JSON-RPC wire format are recognised and, depending on your client library, you can use either of them.

JSON-RPC v1.0

JSON-RPC v1.0 Requests An API call is represented by sending a single JSON object to the server, which
contains the members `method`, `params`, and `id`.

- `method`: A JSON `string` containing the name of the function to be invoked.

- `params`: A JSON `array` of values, which represents the parameters of the function to be invoked.
- `id`: A JSON `string` or `integer` representing the call id. Note that, diverging from the JSON-RPC v1.0 specification the API does not accept *notification* requests (requests without responses), i.e. the id cannot be **`null`**.

For example, a JSON-RPC v1.0 request to retrieve the resident VMs of a host may look like this:

```
  {

    "method": "host.get_resident_VMs",
    "params": [
      "OpaqueRef:74f1a19cd-b660-41e3-a163-10f03e0eae67",
      "OpaqueRef:08c34fc9-f418-4f09-8274-b9cb25cd8550"
    ],
    "id": "xyz"
   }

<!--NeedCopy-->
```

In the above example, the first element of the `params` array is the reference of the open session to the host, while the second is the host reference.

JSON-RPC v1.0 Return Values The return value of a JSON-RPC v1.0 call is a single JSON object containing
the members `result`, `error`, and `id`.

- `result`: If the call is successful, it is a JSON value (`string`, `array` etc.) representing the return value of the invoked function. If an error has occurred, it is **`null`**.
- `error`: If the call is successful, it is **`null`**. If the call has failed, it a JSON `array` of `string` values. The first element of the array is an error code; the remainder of the array are strings representing error parameters relating to that code.
- `id`: The call id. It is a JSON `string` or `integer` and it is the same id as the request it is responding to.

For example, a JSON-RPC v1.0 return value from the `host.get_resident_VMs` function may look like this:

```
  {

    "result": [
        "OpaqueRef:604f51e7-630f-4412-83fa-b11c6cf008ab",
        "OpaqueRef:670d08f5-cbeb-4336-8420-ccd56390a65f"
    ],
    "error": null,
    "id": "xyz"
  }

<!--NeedCopy-->
```

while the return value of the same call made on a logged out session may look
like this:

```
  {

    "result": null,
    "error": [
        "SESSION_INVALID",
        "OpaqueRef:93f1a23cd-a640-41e3-b163-10f86e0eae67"
    ],
    "id": "xyz"
  }

<!--NeedCopy-->
```

JSON-RPC v2.0

JSON-RPC v2.0 Requests An API call is represented by sending a single JSON object to the server, which
contains the members `jsonrpc`, `method`, `params`, and `id`.

- `jsonrpc`: A JSON `string` specifying the version of the JSON-RPC protocol. It is exactly "2.0".
- `method`: A JSON `string` containing the name of the function to be invoked.
- `params`: A JSON `array` of values, which represents the parameters of the function to be invoked. Although the JSON-RPC v2.0 specification allows this member to be ommitted, in practice all API calls accept at least one parameter.
- `id`: A JSON `string` or `integer` representing the call id. Note that, diverging from the JSON-RPC v2.0 specification it cannot be null. Neither can it be ommitted because the API does not accept *notification* requests (requests without responses).

For example, a JSON-RPC v2.0 request to retrieve the VMs resident on a host may
may look like this:

```
{

  "jsonrpc": "2.0",
  "method": "host.get_resident_VMs",
  "params": [
    "OpaqueRef:c90cd28f-37ec-4dbf-88e6-f697ccb28b39",
    "OpaqueRef:08c34fc9-f418-4f09-8274-b9cb25cd8550"
  ],
  "id": 3
}

<!--NeedCopy-->
```

As before, the first element of the `parameter` array is the reference
of the open session to the host, while the second is the host reference.

JSON-RPC v2.0 Return Values The return value of a JSON-RPC v2.0 call is a single JSON object containing the
members `jsonrpc`, either `result` or `error` depending on the outcome of the
call, and `id`.

- `jsonrpc`: A JSON `string` specifying the version of the JSON-RPC protocol. It is exactly "2.0".
- `result`: If the call is successful, it is a JSON value (`string`, `array` etc.) representing the return value of the invoked function. If an error has occurred, it does not exist.
- `error`: If the call is successful, it does not exist. If the call has failed, it is a single structured JSON object (see below).
- `id`: The call id. It is a JSON `string` or `integer` and it is the same id as the request it is responding to.

The `error` object contains the members `code`, `message`, and `data`.

- `code`: The API does not make use of this member and only retains it for compliance with the JSON-RPC v2.0 specification. It is a JSON `integer` which has a non-zero value.
- `message`: A JSON `string` representing an API error code.
- `data`: A JSON array of `string` values representing error parameters relating to the aforementioned API error code.

For example, a JSON-RPC v2.0 return value from the `host.get_resident_VMs`
function may look like this:

```
  {

    "jsonrpc": "2.0",
    "result": [
        "OpaqueRef:604f51e7-630f-4412-83fa-b11c6cf008ab",
        "OpaqueRef:670d08f5-cbeb-4336-8420-ccd56390a65f"
    ],
    "id": 3
   }

<!--NeedCopy-->
```

while the return value of the same call made on a logged out session may look like this:

```
  {

    "jsonrpc": "2.0",
    "error": {

        "code": 1,
        "message": "SESSION_INVALID",
        "data": [
            "OpaqueRef:c90cd28f-37ec-4dbf-88e6-f697ccb28b39"
        ]
      }
 ,
    "id": 3
   }

<!--NeedCopy-->
```

Note on References vs UUIDs

References are opaque types - encoded as XML-RPC and JSON-RPC strings on the wire - understood only by the particular server which generated them. Servers are free to choose any concrete representation they find convenient; clients should not make any assumptions or attempt to parse the string contents. References are not guaranteed to be permanent identifiers for objects; clients should not assume that references generated during one session are valid for any future session. References do not allow objects to be compared for equality. Two references to the same object are not guaranteed to be textually identical.

UUIDs are intended to be permanent names for objects. They are guaranteed to be in the OSF DCE UUID presentation format (as output by `uuidgen`). Clients may store UUIDs on disk and use them to lookup objects in subsequent sessions with the server. Clients may also test equality on objects by comparing UUID strings.

The API provides mechanisms for translating between UUIDs and opaque references.
Each class that contains a UUID field provides:

- A `get_by_uuid` method that takes a UUID and returns an opaque reference to the server-side object that has that UUID;
- A `get_uuid` function (a regular "field getter"RPC) that takes an opaque reference and returns the UUID of the server-side object that is referenced by it.

Making RPC Calls

Transport Layer

The following transport layers are currently supported:

- HTTP/HTTPS for remote administration
- HTTP over Unix domain sockets for local administration

Session Layer

The RPC interface is session-based; before you can make arbitrary RPC calls
you must login and initiate a session. For example:

```
    (session ref) session.login_with_password(string uname, string pwd,
                        string version, string originator)
<!--NeedCopy-->
```

where `uname` and `password` refer to your username and password, as defined by the Xen administrator, while `version` and `originator` are optional. The `session ref` returned by `session.login_with_password` is passed to subequent RPC calls as an authentication token. Note that a session reference obtained by a login request to the XML-RPC backend can be used in subsequent requests to the JSON-RPC backend, and vice-versa.

A session can be terminated with the `session.logout` function:

```
    void  session.logout(session ref session_id)
<!--NeedCopy-->
```

Synchronous and Asynchronous Invocation

Each method call (apart from methods on the `Session` and `Task` objects and "getters"and "setters"derived from fields) can be made either synchronously or

asynchronously. A synchronous RPC call blocks until the
return value is received; the return value of a synchronous RPC call is
exactly as specified above.

Only synchronous API calls are listed explicitly in this document.
All their asynchronous counterparts are in the special `Async` namespace.
For example, the synchronous call `VM.clone(...)` has an asynchronous
counterpart, `Async.VM.clone(...)`, that is non-blocking.

Instead of returning its result directly, an asynchronous RPC call
returns an identifier of type `task ref` which is subsequently used
to track the status of a running asynchronous RPC.

Note that an asychronous call may fail immediately, before a task has even been
created. When using the XML-RPC wire protocol, this eventuality is represented
by wrapping the returned `task ref` in an XML-RPC struct with a `Status`,
`ErrorDescription`, and `Value` fields, exactly as specified above; the
`task ref` is provided in the `Value` field if `Status` is set to `Success`.
When using the JSON-RPC protocol, the `task ref` is wrapped in a response JSON
object as specified above and it is provided by the value of the `result` member
of a successful call.

The RPC call

```
    (task ref set)  Task.get_all(session ref session_id)
<!--NeedCopy-->
```

returns a set of all task identifiers known to the system. The status (including any
returned result and error codes) of these can then be queried by accessing the
fields of the `Task` object in the usual way. Note that, in order to get a
consistent snapshot of a task's state, it is advisable to call the `get_record`
function.

Example interactive session

This section describes how an interactive session might look, using python
XML-RPC and JSON-RPC client libraries.

First, initialise python:

```
$ python2.7
>>>
<!--NeedCopy-->
```

Using the XML-RPC Protocol

Import the library `xmlrpclib` and create a
python object referencing the remote server as shown below:

```
>>> import xmlrpclib
>>> xen = xmlrpclib.Server("https://localhost:443")
<!--NeedCopy-->
```

Acquire a session reference by logging in with a username and password; the
session reference is returned under the key `Value` in the resulting dictionary
(error-handling ommitted for brevity):

```
>>> session = xen.session.login_with_password("user", "passwd",
...                                            "version", "originator")[
    'Value']
<!--NeedCopy-->
```

This is what the call looks like when serialised

```
<?xml version='1.0'?>
<methodCall>
    <methodName>session.login_with_password</methodName>
    <params>
        <param><value><string>user</string></value></param>
        <param><value><string>passwd</string></value></param>
        <param><value><string>version</string></value></param>
        <param><value><string>originator</string></value></param>
    </params>
</methodCall>
<!--NeedCopy-->
```

Next, the user may acquire a list of all the VMs known to the system (note the
call takes the session reference as the only parameter):

```
>>> all_vms = xen.VM.get_all(session)['Value']
>>> all_vms
['OpaqueRef:1', 'OpaqueRef:2', 'OpaqueRef:3', 'OpaqueRef:4' ]
<!--NeedCopy-->
```

The VM references here have the form `OpaqueRef:X` (though they may not be
that simple in reality) and you should treat them as opaque strings.
Templates are VMs with the `is_a_template` field set to **`true`**. We can
find the subset of template VMs using a command like the following:

```
>>> all_templates = filter(lambda x: xen.VM.get_is_a_template(session,
    x)['Value'],
                              all_vms)
<!--NeedCopy-->
```

Once a reference to a VM has been acquired, a lifecycle operation may be invoked:

```
>>> xen.VM.start(session, all_templates[0], False, False)
{
 'Status': 'Failure', 'ErrorDescription': ['VM_IS_TEMPLATE', 'OpaqueRef
    :X'] }

<!--NeedCopy-->
```

In this case the `start` message has been rejected, because the VM is
a template, and so an error response has been returned. These high-level
errors are returned as structured data (rather than as XML-RPC faults),
allowing them to be internationalised.

Rather than querying fields individually, whole *records* may be returned at once.
To retrieve the record of a single object as a python dictionary:

```
>>> record = xen.VM.get_record(session, all_templates[0])['Value']
>>> record['power_state']
'Halted'
>>> record['name_label']
'Windows 10 (64-bit)'
<!--NeedCopy-->
```

To retrieve all the VM records in a single call:

```
>>> records = xen.VM.get_all_records(session)['Value']
>>> records.keys()
['OpaqueRef:1', 'OpaqueRef:2', 'OpaqueRef:3', 'OpaqueRef:4' ]
>>> records['OpaqueRef:1']['name_label']
'Red Hat Enterprise Linux 7'
<!--NeedCopy-->
```

Using the JSON-RPC Protocol

For this example we are making use of the package `python-jsonrpc` due to its
simplicity, although other packages can also be used.

First, import the library `pyjsonrpc` and create the object referencing the
remote server as follows:

```
>>> import pyjsonrpc
>>> client = pyjsonrpc.HttpClient(url = "https://localhost/jsonrpc:443"
    )
<!--NeedCopy-->
```

Acquire a session reference by logging in with a username and password; the
library `pyjsonrpc` returns the response's `result` member, which is the session
reference:

```
>>> session = client.call("session.login_with_password",
...                        "user", "passwd", "version", "originator")
<!--NeedCopy-->
```

pyjsonrpc uses the JSON-RPC protocol v2.0, so this is what the serialised request looks like:

```
  {

    "jsonrpc": "2.0",
    "method": "session.login_with_password",
    "params": ["user", "passwd", "version", "originator"],
    "id": 0
   }

<!--NeedCopy-->
```

Next, the user may acquire a list of all the VMs known to the system (note the call takes the session reference as the only parameter):

```
>>> all_vms = client.call("VM.get_all", session)
>>> all_vms
['OpaqueRef:1', 'OpaqueRef:2', 'OpaqueRef:3', 'OpaqueRef:4' ]
<!--NeedCopy-->
```

The VM references here have the form `OpaqueRef:X` (though they may not be that simple in reality) and you should treat them as opaque strings. *Templates* are VMs with the `is_a_template` field set to **`true`**. We can find the subset of template VMs using a command like the following:

```
>>> all_templates = filter(
...     lambda x: client.call("VM.get_is_a_template", session, x),
        all_vms)
<!--NeedCopy-->
```

Once a reference to a VM has been acquired, a lifecycle operation may be invoked:

```
>>> from pyjsonrpc import JsonRpcError
>>> try:
...     client.call("VM.start", session, all_templates[0], False, False
    )
... except JsonRpcError as e:
...     e.message
...     e.data
...
'VM_IS_TEMPLATE'
[ 'OpaqueRef:1', 'start' ]
<!--NeedCopy-->
```

In this case the `start` message has been rejected because the VM is

a template, hence an error response has been returned. These high-level
errors are returned as structured data, allowing them to be internationalised.

Rather than querying fields individually, whole *records* may be returned at once.
To retrieve the record of a single object as a python dictionary:

```
>>> record = client.call("VM.get_record", session, all_templates[0])
>>> record['power_state']
'Halted'
>>> record['name_label']
'Windows 10 (64-bit)'
<!--NeedCopy-->
```

To retrieve all the VM records in a single call:

```
>>> records = client.call("VM.get_all_records", session)
>>> records.keys()
['OpaqueRef:1', 'OpaqueRef:2', 'OpaqueRef:3', 'OpaqueRef:4' ]
>>> records['OpaqueRef:1']['name_label']
'Red Hat Enterprise Linux 7'
<!--NeedCopy-->
```

VM Lifecycle

January 23, 2024

The following diagram shows the states that a VM can be in
and the API calls that can be used to move the VM between these states.

VM boot parameters

The `VM` class contains a number of fields that control the way in which the VM
is booted. With reference to the fields defined in the VM class (see later in
this document), this section outlines the boot options available and the
mechanisms provided for controlling them.

VM booting is controlled by setting one of the two mutually exclusive groups:
“PV”and “HVM”. If `HVM.boot_policy` is an empty string, then paravirtual
domain building and booting will be used; otherwise the VM will be loaded as a
HVM domain, and booted using an emulated BIOS.

When paravirtual booting is in use, the `PV_bootloader` field indicates the
bootloader to use. It may be “pygrub”, in which case the platform’s default
installation of pygrub will be used, or a full path within the control domain to

some other bootloader. The other fields, PV_kernel, PV_ramdisk, PV_args, and PV_bootloader_args will be passed to the bootloader unmodified, and interpretation of those fields is then specific to the bootloader itself, including the possibility that the bootloader will ignore some or all of those given values. Finally the paths of all bootable disks are added to the bootloader commandline (a disk is bootable if its VBD has the bootable flag set). There may be zero, one, or many bootable disks; the bootloader decides which disk (if any) to boot from.

If the bootloader is pygrub, then the menu.lst is parsed, if present in the guest's filesystem, otherwise the specified kernel and ramdisk are used, or an autodetected kernel is used if nothing is specified and autodetection is possible. PV_args is appended to the kernel command line, no matter which mechanism is used for finding the kernel.

If PV_bootloader is empty but PV_kernel is specified, then the kernel and ramdisk values will be treated as paths within the control domain. If both PV_bootloader and PV_kernel are empty, then the behaviour is as if PV_bootloader were specified as "pygrub".

When using HVM booting, HVM_boot_policy and HVM_boot_params specify the boot handling. Only one policy is currently defined, "BIOS order". In this case, HVM_boot_params should contain one key-value pair "order"= "N"where N is the string that will be passed to QEMU.
Optionally HVM_boot_params can contain another key-value pair "firmware" with values "bios"or "uefi"(default is "bios"if absent).
By default Secure Boot is not enabled, it can be enabled when "uefi"is enabled by setting VM.platform["secureboot"] to true.

API Reference - Types and Classes

January 23, 2024

Classes

The following classes are defined:

Name	Description
`auth`	Management of remote authentication services
`blob`	A placeholder for a binary blob
`Bond`	
`Certificate`	Description
`Cluster`	Cluster-wide Cluster metadata
`Cluster_host`	Cluster member metadata
`console`	A console
`crashdump`	**Deprecated**. A VM crashdump
`data_source`	Data sources for logging in RRDs
`DR_task`	DR task
`event`	Asynchronous event registration and handling
`Feature`	A new piece of functionality
`GPU_group`	A group of compatible GPUs across the resource pool
`host`	A physical host
`host_cpu`	**Deprecated**. A physical CPU
`host_crashdump`	Represents a host crash dump
`host_metrics`	The metrics associated with a host
`host_patch`	**Deprecated**. Represents a patch stored on a server
`LVHD`	LVHD SR specific operations
`message`	An message for the attention of the administrator
`network`	A virtual network
`network_sriov`	network-sriov which connects logical pif and physical pif
`Observer`	Describes a observer which will control observability activity in the Toolstack
`PBD`	The physical block devices through which hosts access SRs
`PCI`	A PCI device

Name	Description
`PGPU`	A physical GPU (pGPU)
`PIF`	A physical network interface (note separate VLANs are represented as several PIFs)
`PIF_metrics`	The metrics associated with a physical network interface
`pool`	Pool-wide information
`pool_patch`	**Deprecated**. Pool-wide patches
`pool_update`	Pool-wide updates to the host software
`probe_result`	A set of properties that describe one result element of SR.probe. Result elements and properties can change dynamically based on changes to the the SR.probe input-parameters or the target.
`PUSB`	A physical USB device
`PVS_cache_storage`	Describes the storage that is available to a PVS site for caching purposes
`PVS_proxy`	a proxy connects a VM/VIF with a PVS site
`PVS_server`	individual machine serving provisioning (block) data
`PVS_site`	machines serving blocks of data for provisioning VMs
`Repository`	Repository for updates
`role`	A set of permissions associated with a subject
`SDN_controller`	Describes the SDN controller that is to connect with the pool
`secret`	A secret
`session`	A session
`SM`	A storage manager plugin
`SR`	A storage repository
`sr_stat`	A set of high-level properties associated with an SR.
`subject`	A user or group that can log in xapi
`task`	A long-running asynchronous task

Name	Description
`tunnel`	A tunnel for network traffic
`USB_group`	A group of compatible USBs across the resource pool
`user`	**Deprecated**. A user of the system
`VBD`	A virtual block device
`VBD_metrics`	**Removed**. The metrics associated with a virtual block device
`VDI`	A virtual disk image
`vdi_nbd_server_info`	Details for connecting to a VDI using the Network Block Device protocol
`VGPU`	A virtual GPU (vGPU)
`VGPU_type`	A type of virtual GPU
`VIF`	A virtual network interface
`VIF_metrics`	**Removed**. The metrics associated with a virtual network device
`VLAN`	A VLAN mux/demux
`VM`	A virtual machine (or 'guest').
`VM_appliance`	VM appliance
`VM_guest_metrics`	The metrics reported by the guest (as opposed to inferred from outside)
`VM_metrics`	The metrics associated with a VM
`VMPP`	**Removed**. VM Protection Policy
`VMSS`	VM Snapshot Schedule
`VTPM`	A virtual TPM device
`VUSB`	Describes the vusb device

Relationships Between Classes

Fields that are bound together are shown in the following table:

object.field	*object.field*	*relationship*
`VM.snapshot_of`	`VM.snapshots`	one-to-many
`VDI.snapshot_of`	`VDI.snapshots`	one-to-many
`VM.parent`	`VM.children`	one-to-many
`task.subtask_of`	`task.subtasks`	one-to-many
`PIF.bond_slave_of`	`Bond.slaves`	one-to-many
`Bond.master`	`PIF.bond_master_of`	one-to-many
`VLAN.tagged_PIF`	`PIF.VLAN_slave_of`	one-to-many
`tunnel.access_PIF`	`PIF.tunnel_access_PIF_of`	one-to-many
`tunnel.transport_PIF`	`PIF.tunnel_transport_PIF_of`	one-to-many
`PBD.host`	`host.PBDs`	one-to-many
`PBD.SR`	`SR.PBDs`	one-to-many
`VBD.VDI`	`VDI.VBDs`	one-to-many
`crashdump.VDI`	`VDI.crash_dumps`	one-to-many
`VBD.VM`	`VM.VBDs`	one-to-many
`crashdump.VM`	`VM.crash_dumps`	one-to-many
`VIF.VM`	`VM.VIFs`	one-to-many
`VIF.network`	`network.VIFs`	one-to-many
`Cluster_host.cluster`	`Cluster.cluster_hosts`	one-to-many
`PIF.host`	`host.PIFs`	one-to-many
`PIF.network`	`network.PIFs`	one-to-many
`VDI.SR`	`SR.VDIs`	one-to-many
`VTPM.VM`	`VM.VTPMs`	one-to-many
`console.VM`	`VM.consoles`	one-to-many
`VM.resident_on`	`host.resident_VMs`	one-to-many
`host_cpu.host`	`host.host_CPUs`	one-to-many
`host_crashdump.host`	`host.crashdumps`	one-to-many

object.field	*object.field*	*relationship*
host_patch.host	host.patches	one-to-many
host_patch.pool_patch	pool_patch.host_patches	one-to-many
host.updates	pool_update.hosts	many-to-many
subject.roles	subject.roles	unknown type
role.subroles	role.subroles	many-to-many
VM.protection_policy	VMPP.VMs	one-to-many
VM.snapshot_schedule	VMSS.VMs	one-to-many
VM.appliance	VM_appliance.VMs	one-to-many
PGPU.GPU_group	GPU_group.PGPUs	one-to-many
VGPU.GPU_group	GPU_group.VGPUs	one-to-many
VGPU.type	VGPU_type.VGPUs	one-to-many
VGPU.VM	VM.VGPUs	one-to-many
VGPU.resident_on	PGPU.resident_VGPUs	one-to-many
PGPU.supported_VGPU_types	VGPU_type.supported_on_PGPUs	many-to-many
PGPU.enabled_VGPU_types	VGPU_type.enabled_on_PGPUs	many-to-many
GPU_group.supported_VGPU_types	VGPU_type.supported_on_GPU_groups	many-to-many
GPU_group.enabled_VGPU_types	VGPU_type.enabled_on_GPU_groups	many-to-many
PCI.host	host.PCIs	one-to-many
PGPU.host	host.PGPUs	one-to-many
VDI.metadata_of_pool	pool.metadata_VDIs	one-to-many
SR.introduced_by	DR_task.introduced_SRs	one-to-many
PVS_server.site	PVS_site.servers	one-to-many
PVS_proxy.site	PVS_site.proxies	one-to-many

object.field	*object.field*	*relationship*
`PVS_cache_storage.site`	`PVS_site.cache_storage`	one-to-many
`PUSB.host`	`host.PUSBs`	one-to-many
`PUSB.USB_group`	`USB_group.PUSBs`	one-to-many
`VUSB.USB_group`	`USB_group.VUSBs`	one-to-many
`VUSB.VM`	`VM.VUSBs`	one-to-many
`Feature.host`	`host.features`	one-to-many
`network_sriov.physical_PIF`	`PIF.sriov_physical_PIF_of`	one-to-many
`network_sriov.logical_PIF`	`PIF.sriov_logical_PIF_of`	one-to-many
`Certificate.host`	`host.certificates`	one-to-many

The following figure represents bound fields (as specified above) diagramatically, using crow's foot notation to specify one-to-one, one-to-many or many-to-many relationships:

Types

Primitives

The following primitive types are used to specify methods and fields in the API Reference:

Type	Description
string	text strings
int	64-bit integers
float	IEEE double-precision floating-point numbers
bool	boolean
datetime	date and timestamp

Higher-order types

The following type constructors are used:

Type	Description
c ref	reference to an object of class *c*
t set	a set of elements of type *t*
(*a* -> *b*) map	a table mapping values of type *a* to values of type *b*

Enumeration types

The following enumeration types are used:

`enum after_apply_guidance`	
`restartHost`	This patch requires the host to be restarted once applied.
`restartHVM`	This patch requires HVM guests to be restarted once applied.
`restartPV`	This patch requires PV guests to be restarted once applied.
`restartXAPI`	This patch requires XAPI to be restarted once applied.

`enum allocation_algorithm`	
`breadth_first`	vGPUs of a given type are allocated evenly across supporting pGPUs.
`depth_first`	vGPUs of a given type are allocated on supporting pGPUs until they are full.

`enum bond_mode`	
`active-backup`	Active/passive bonding: only one NIC is carrying traffic
`balance-slb`	Source-level balancing
`lacp`	Link aggregation control protocol

enum certificate_type	
ca	Certificate that is trusted by the whole pool
host	Certificate that identifies a single host to entities outside the pool
host_internal	Certificate that identifies a single host to other pool members

enum cls	
Certificate	Certificate
Host	Host
Pool	Pool
PVS_proxy	PVS_proxy
SR	SR
VDI	VDI
VM	VM
VMPP	VMPP
VMSS	VMSS

enum cluster_host_operation	
destroy	completely destroying a cluster host
disable	disabling cluster membership on a particular host
enable	enabling cluster membership on a particular host

enum cluster_operation	
add	adding a new member to the cluster
destroy	completely destroying a cluster
disable	disabling any cluster member

`enum cluster_operation`	
`enable`	enabling any cluster member
`remove`	removing a member from the cluster

`enum console_protocol`	
`rdp`	Remote Desktop Protocol
`rfb`	Remote FrameBuffer protocol (as used in VNC)
`vt100`	VT100 terminal

`enum domain_type`	
`hvm`	HVM; Fully Virtualised
`pv`	PV: Paravirtualised
`pv_in_pvh`	PV inside a PVH container
`pvh`	PVH
`unspecified`	Not specified or unknown domain type

`enum event_operation`	
`add`	An object has been created
`del`	An object has been deleted
`mod`	An object has been modified

`enum host_allowed_operations`	
`apply_updates`	Indicates this host is being updated
`evacuate`	Indicates this host is evacuating
`power_on`	Indicates this host is in the process of being powered on
`provision`	Indicates this host is able to provision another VM

enum host_allowed_operations	
reboot	Indicates this host is in the process of rebooting
shutdown	Indicates this host is in the process of shutting itself down
vm_migrate	This host is the migration target of a VM
vm_resume	This host is resuming a VM
vm_start	This host is starting a VM

enum host_display	
disable_on_reboot	The host will stop outputting its console to a physical display device on next boot
disabled	This host is not outputting its console to a physical display device
enable_on_reboot	The host will start outputting its console to a physical display device on next boot
enabled	This host is outputting its console to a physical display device

enum host_numa_affinity_policy	
any	VMs are spread across all available NUMA nodes
best_effort	VMs are placed on the smallest number of NUMA nodes that they fit using soft-pinning, but the policy doesn't guarantee a balanced placement, falling back to the 'any'policy.
default_policy	Use the NUMA affinity policy that is the default for the current version

enum host_sched_gran	
core	core scheduling
cpu	CPU scheduling
socket	socket scheduling

enum ip_configuration_mode	
DHCP	Acquire an IP address by DHCP
None	Do not acquire an IP address
Static	Static IP address configuration

enum ipv6_configuration_mode	
Autoconf	Router assigned prefix delegation IPv6 allocation
DHCP	Acquire an IPv6 address by DHCP
None	Do not acquire an IPv6 address
Static	Static IPv6 address configuration

enum latest_synced_updates_applied_state	
no	The host is outdated with the latest updates synced from remote CDN
unknown	If the host is up to date with the latest updates synced from remote CDN is unknown
yes	The host is up to date with the latest updates synced from remote CDN

enum livepatch_status	
ok	There is no applicable live patch
ok_livepatch_complete	An applicable live patch exists for every required component
ok_livepatch_incomplete	An applicable live patch exists but it is not sufficient

enum network_default_locking_mode	
disabled	Treat all VIFs on this network with locking_mode = 'default'as if they have locking_mode = 'disabled'
unlocked	Treat all VIFs on this network with locking_mode = 'default'as if they have locking_mode = 'unlocked'

enum network_operations	
attaching	Indicates this network is attaching to a VIF or PIF

enum network_purpose	
insecure_nbd	Network Block Device service without integrity or confidentiality: NOT RECOMMENDED
nbd	Network Block Device service using TLS

enum on_boot	
persist	Standard behaviour.
reset	When a VM containing this VDI is started, the contents of the VDI are reset to the state they were in when this flag was last set.

enum on_crash_behaviour	
coredump_and_destroy	record a coredump and then destroy the VM state
coredump_and_restart	record a coredump and then restart the VM
destroy	destroy the VM state
preserve	leave the crashed VM paused
rename_restart	rename the crashed VM and start a new copy
restart	restart the VM

enum on_normal_exit	
destroy	destroy the VM state
restart	restart the VM

enum on_softreboot_behavior	
destroy	destroy the VM state
preserve	leave the VM paused
restart	restart the VM
soft_reboot	perform soft-reboot

enum persistence_backend	
xapi	This VTPM is persisted in XAPI's DB

enum pgpu_dom0_access	
disable_on_reboot	On host reboot dom0 will be blocked from accessing this device
disabled	dom0 cannot access this device
enable_on_reboot	On host reboot dom0 will be allowed to access this device
enabled	dom0 can access this device as normal

enum pif_igmp_status	
disabled	IGMP Snooping is disabled in the corresponding backend bridge.'
enabled	IGMP Snooping is enabled in the corresponding backend bridge.'
unknown	IGMP snooping status is unknown. If this is a VLAN master, then please consult the underlying VLAN slave PIF.

enum pool_allowed_operations	
apply_updates	Indicates this pool is in the process of applying updates
cert_refresh	A certificate refresh and distribution is in progress
cluster_create	Indicates this pool is in the process of creating a cluster
configure_repositories	Indicates this pool is in the process of configuring repositories
copy_primary_host_certs	Indicates the primary host is sending its certificates to another host
designate_new_master	Indicates this pool is in the process of changing master
exchange_ca_certificates_on_join	Indicates this pool is exchanging ca certificates with a new joiner
exchange_certificates_on_join	Indicates this pool is exchanging internal certificates with a new joiner
get_updates	Indicates this pool is in the process of getting updates
ha_disable	Indicates this pool is in the process of disabling HA
ha_enable	Indicates this pool is in the process of enabling HA
sync_updates	Indicates this pool is in the process of syncing updates
tls_verification_enable	Indicates this pool is in the process of enabling TLS verification

enum primary_address_type	
IPv4	Primary address is the IPv4 address
IPv6	Primary address is the IPv6 address

enum pvs_proxy_status	
caching	The proxy is currently caching data

enum pvs_proxy_status	
incompatible_protocol_version	The PVS protocol in use is not compatible with the PVS proxy
incompatible_write_cache_mode	The PVS device is configured to use an incompatible write-cache mode
initialised	The proxy is setup but has not yet cached anything
stopped	The proxy is not currently running

enum sdn_controller_protocol	
pssl	Passive ssl connection
ssl	Active ssl connection

enum sr_health	
healthy	Storage is fully available
recovering	Storage is busy recovering, e.g. rebuilding mirrors.

enum sriov_configuration_mode	
manual	Configure network sriov manually
modprobe	Configure network sriov by modprobe, need reboot
sysfs	Configure network sriov by sysfs, do not need reboot
unknown	Unknown mode

enum storage_operations	
destroy	Destroying the SR
forget	Forgetting about SR

enum storage_operations	
pbd_create	Creating a PBD for this SR
pbd_destroy	Destroying one of this SR's PBDs
plug	Plugging a PBD into this SR
scan	Scanning backends for new or deleted VDIs
unplug	Unplugging a PBD from this SR
update	Refresh the fields on the SR
vdi_clone	Cloneing a VDI
vdi_create	Creating a new VDI
vdi_data_destroy	Deleting the data of the VDI
vdi_destroy	Destroying a VDI
vdi_disable_cbt	Disabling changed block tracking for a VDI
vdi_enable_cbt	Enabling changed block tracking for a VDI
vdi_introduce	Introducing a new VDI
vdi_list_changed_blocks	Exporting a bitmap that shows the changed blocks between two VDIs
vdi_mirror	Mirroring a VDI
vdi_resize	Resizing a VDI
vdi_set_on_boot	Setting the on_boot field of the VDI
vdi_snapshot	Snapshotting a VDI

enum task_allowed_operations	
cancel	refers to the operation "cancel"
destroy	refers to the operation "destroy"

enum task_status_type	
cancelled	task has been cancelled
cancelling	task is being cancelled
failure	task has failed

enum task_status_type	
pending	task is in progress
success	task was completed successfully

enum telemetry_frequency	
daily	Run telemetry task daily
monthly	Run telemetry task monthly
weekly	Run telemetry task weekly

enum tristate_type	
no	Known to be false
unspecified	Unknown or unspecified
yes	Known to be true

enum tunnel_protocol	
gre	GRE protocol
vxlan	VxLAN Protocol

enum update_after_apply_guidance	
restartHost	This update requires the host to be restarted once applied.
restartHVM	This update requires HVM guests to be restarted once applied.
restartPV	This update requires PV guests to be restarted once applied.
restartXAPI	This update requires XAPI to be restarted once applied.

enum update_guidances	
reboot_host	Indicates the updated host should reboot as soon as possible
reboot_host_on_livepatch_failure	Indicates the updated host should reboot as soon as possible since one or more livepatch(es) failed to be applied.
restart_device_model	Indicates the device model of a running VM should restart as soon as possible
restart_toolstack	Indicates the Toolstack running on the updated host should restart as soon as possible

enum update_sync_frequency	
daily	The update synchronizations happen every day
weekly	The update synchronizations happen every week on the chosen day

enum vbd_mode	
RO	only read-only access will be allowed
RW	read-write access will be allowed

enum vbd_operations	
attach	Attempting to attach this VBD to a VM
eject	Attempting to eject the media from this VBD
insert	Attempting to insert new media into this VBD
pause	Attempting to pause a block device backend
plug	Attempting to hotplug this VBD
unpause	Attempting to unpause a block device backend
unplug	Attempting to hot unplug this VBD
unplug_force	Attempting to forcibly unplug this VBD

enum vbd_type	
CD	VBD will appear to guest as CD
Disk	VBD will appear to guest as disk
Floppy	VBD will appear as a floppy

enum vdi_operations	
blocked	Operations on this VDI are temporarily blocked
clone	Cloning the VDI
copy	Copying the VDI
data_destroy	Deleting the data of the VDI
destroy	Destroying the VDI
disable_cbt	Disabling changed block tracking for a VDI
enable_cbt	Enabling changed block tracking for a VDI
force_unlock	Forcibly unlocking the VDI
forget	Forget about the VDI
generate_config	Generating static configuration
list_changed_blocks	Exporting a bitmap that shows the changed blocks between two VDIs
mirror	Mirroring the VDI
resize	Resizing the VDI
resize_online	Resizing the VDI which may or may not be online
set_on_boot	Setting the on_boot field of the VDI
snapshot	Snapshotting the VDI
update	Refreshing the fields of the VDI

enum vdi_type	
cbt_metadata	Metadata about a snapshot VDI that has been deleted: the set of blocks that changed between some previous version of the disk and the version tracked by the snapshot.
crashdump	a disk that stores VM crashdump information
ephemeral	a disk that may be reformatted on upgrade
ha_statefile	a disk used for HA storage heartbeating
metadata	a disk used for HA Pool metadata
pvs_cache	a disk that stores PVS cache data
redo_log	a disk used for a general metadata redo-log
rrd	a disk that stores SR-level RRDs
suspend	a disk that stores a suspend image
system	a disk that may be replaced on upgrade
user	a disk that is always preserved on upgrade

enum vgpu_type_implementation	
gvt_g	vGPU using Intel GVT-g
mxgpu	vGPU using AMD MxGPU
nvidia	vGPU using NVIDIA hardware
nvidia_sriov	vGPU using NVIDIA hardware with SR-IOV
passthrough	Pass through an entire physical GPU to a guest

enum vif_ipv4_configuration_mode	
None	Follow the default IPv4 configuration of the guest (this is guest-dependent)
Static	Static IPv4 address configuration

enum vif_ipv6_configuration_mode	
None	Follow the default IPv6 configuration of the guest (this is guest-dependent)
Static	Static IPv6 address configuration

enum vif_locking_mode	
disabled	No traffic is permitted
locked	Only traffic to a specific MAC and a list of IPv4 or IPv6 addresses is permitted
network_default	No specific configuration set - default network policy applies
unlocked	All traffic is permitted

enum vif_operations	
attach	Attempting to attach this VIF to a VM
plug	Attempting to hotplug this VIF
unplug	Attempting to hot unplug this VIF

enum vm_appliance_operation	
clean_shutdown	Clean shutdown
hard_shutdown	Hard shutdown
shutdown	Shutdown
start	Start

enum vm_operations	
assert_operation_valid	
awaiting_memory_live	Waiting for the memory settings to change
call_plugin	refers to the operation “call_plugin”

`enum vm_operations`	
`changing_dynamic_range`	Changing the memory dynamic range
`changing_memory_limits`	Changing the memory limits
`changing_memory_live`	Changing the memory settings
`changing_NVRAM`	Changing NVRAM for a halted VM.
`changing_shadow_memory`	Changing the shadow memory for a halted VM.
`changing_shadow_memory_live`	Changing the shadow memory for a running VM.
`changing_static_range`	Changing the memory static range
`changing_VCPUs`	Changing VCPU settings for a halted VM.
`changing_VCPUs_live`	Changing VCPU settings for a running VM.
`checkpoint`	refers to the operation “checkpoint”
`clean_reboot`	refers to the operation “clean_reboot”
`clean_shutdown`	refers to the operation “clean_shutdown”
`clone`	refers to the operation “clone”
`copy`	refers to the operation “copy”
`create_template`	refers to the operation “create_template”
`create_vtpm`	Creating and adding a VTPM to this VM
`csvm`	refers to the operation “csvm”
`data_source_op`	Add, remove, query or list data sources
`destroy`	refers to the act of uninstalling the VM
`export`	exporting a VM to a network stream
`get_boot_record`	refers to the operation “get_boot_record”
`hard_reboot`	refers to the operation “hard_reboot”
`hard_shutdown`	refers to the operation “hard_shutdown”
`import`	importing a VM from a network stream
`make_into_template`	Turning this VM into a template
`metadata_export`	exporting VM metadata to a network stream
`migrate_send`	refers to the operation “migrate_send”
`pause`	refers to the operation “pause”
`pool_migrate`	refers to the operation “pool_migrate”

enum vm_operations	
power_state_reset	refers to the operation "power_state_reset"
provision	refers to the operation "provision"
query_services	refers to the operation "query_services"
resume	refers to the operation "resume"
resume_on	refers to the operation "resume_on"
revert	refers to the operation "revert"
reverting	Reverting the VM to a previous snapshotted state
send_sysrq	refers to the operation "send_sysrq"
send_trigger	refers to the operation "send_trigger"
shutdown	refers to the operation "shutdown"
snapshot	refers to the operation "snapshot"
snapshot_with_quiesce	refers to the operation "snapshot_with_quiesce"
start	refers to the operation "start"
start_on	refers to the operation "start_on"
suspend	refers to the operation "suspend"
unpause	refers to the operation "unpause"
update_allowed_operations	

enum vm_power_state	
Halted	VM is offline and not using any resources
Paused	All resources have been allocated but the VM itself is paused and its vCPUs are not running
Running	Running
Suspended	VM state has been saved to disk and it is nolonger running. Note that disks remain in-use while the VM is suspended.

enum vmpp_archive_frequency	
always_after_backup	Archive after backup
daily	Daily archives
never	Never archive
weekly	Weekly backups

enum vmpp_archive_target_type	
cifs	CIFS target config
nfs	NFS target config
none	No target config

enum vmpp_backup_frequency	
daily	Daily backups
hourly	Hourly backups
weekly	Weekly backups

enum vmpp_backup_type	
checkpoint	The backup is a checkpoint
snapshot	The backup is a snapshot

enum vmss_frequency	
daily	Daily snapshots
hourly	Hourly snapshots
weekly	Weekly snapshots

`enum vmss_type`	
`checkpoint`	The snapshot is a checkpoint
`snapshot`	The snapshot is a disk snapshot
`snapshot_with_quiesce`	Support for VSS has been removed.

`enum vtpm_operations`	
`destroy`	Destroy a VTPM

`enum vusb_operations`	
`attach`	Attempting to attach this VUSB to a VM
`plug`	Attempting to plug this VUSB into a VM
`unplug`	Attempting to hot unplug this VUSB

Class: auth

Management of remote authentication services

Fields for class: auth

Class auth has no fields.

RPCs associated with class: auth

RPC name: get_group_membership *Overview:*

This calls queries the external directory service to obtain the transitively-closed set of groups that the the subject_identifier is member of.

Signature:

```
string set get_group_membership (session ref session_id, string
    subject_identifier)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	subject_identifier	A string containing the subject_identifier, unique in the external directory service

Minimum Role: read-only

Return Type: `string set`

set of subject_identifiers that provides the group membership of subject_identifier passed as argument, it contains, recursively, all groups a subject_identifier is member of.

RPC name: get_subject_identifier *Overview:*

This call queries the external directory service to obtain the subject_identifier as a string from the human-readable subject_name

Signature:

```
string get_subject_identifier (session ref session_id, string
    subject_name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	subject_name	The human-readable subject_name, such as a username or a groupname

Minimum Role: read-only

Return Type: `string`

the subject_identifier obtained from the external directory service

RPC name: get_subject_information_from_identifier *Overview:*

This call queries the external directory service to obtain the user information (e.g. username, organization etc) from the specified subject_identifier

Signature:

```
(string -> string) map get_subject_information_from_identifier (session
    ref session_id, string subject_identifier)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	subject_identifier	A string containing the subject_identifier, unique in the external directory service

Minimum Role: read-only

Return Type: `(string -> string)map`

key-value pairs containing at least a key called subject_name

Class: blob

A placeholder for a binary blob

Fields for class: blob

Field	Type	Qualifier	Description
last_updated	`datetime`	*RO/constructor*	Time at which the data in the blob was last updated

Field	Type	Qualifier	Description
mime_type	string	*RO/constructor*	The mime type associated with this object. Defaults to 'application/octet-stream' if the empty string is supplied
name_description	string	*RW*	a notes field containing human-readable description
name_label	string	*RW*	a human-readable name
public	bool	*RW*	True if the blob is publicly accessible
size	int	*RO/runtime*	Size of the binary data, in bytes
uuid	string	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: blob

RPC name: create *Overview:*

Create a placeholder for a binary blob

Signature:

```
blob ref create (session ref session_id, string mime_type, bool public)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`string`	mime_type	The mime-type of the blob. Defaults to 'application/octet-stream' if the empty string is supplied
`bool`	public	True if the blob should be publicly available

Minimum Role: pool-operator

Return Type: `blob ref`

The reference to the created blob

RPC name: destroy *Overview:*

Signature:

```
void destroy (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	The reference of the blob to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the blobs known to the system.

Signature:

```
blob ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `blob ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of blob references to blob records for all blobs known to the system.

Signature:

```
(blob ref -> blob record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(blob ref -> blob record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the blob instances with the given label.

Signature:

```
blob ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `blob ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the blob instance with the specified UUID.

Signature:

```
blob ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `blob ref`

reference to the object

RPC name: get_last_updated *Overview:*

Get the last_updated field of the given blob.

Signature:

```
datetime get_last_updated (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_mime_type *Overview:*

Get the mime_type field of the given blob.

Signature:

```
string get_mime_type (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given blob.

Signature:

```
string get_name_description (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given blob.

Signature:

```
string get_name_label (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_public *Overview:*

Get the public field of the given blob.

Signature:

```
bool get_public (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given blob.

Signature:

```
blob record get_record (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `blob record`

all fields from the object

RPC name: get_size *Overview:*

Get the size field of the given blob.

Signature:

```
int get_size (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given blob.

Signature:

```
string get_uuid (session ref session_id, blob ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: set_name_description *Overview:*

Set the name/description field of the given blob.

Signature:

```
void set_name_description (session ref session_id, blob ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_name_label *Overview:*

Set the name/label field of the given blob.

Signature:

```
void set_name_label (session ref session_id, blob ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_public *Overview:*

Set the public field of the given blob.

Signature:

```
void set_public (session ref session_id, blob ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`blob ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Class: Bond

Fields for class: Bond

Field	Type	Qualifier	Description
auto_update_mac	`bool`	*RO/runtime*	true if the MAC was taken from the primary slave when the bond was created, and false if the client specified the MAC
links_up	`int`	*RO/runtime*	Number of links up in this bond
master	`PIF ref`	*RO/constructor*	The bonded interface
mode	`bond_mode`	*RO/runtime*	The algorithm used to distribute traffic among the bonded NICs
other_config	`(string -> string)map`	*RW*	additional configuration
primary_slave	`PIF ref`	*RO/runtime*	The PIF of which the IP configuration and MAC were copied to the bond, and which will receive all configuration/VLANs/VIFs on the bond if the bond is destroyed
properties	`(string -> string)map`	*RO/runtime*	Additional configuration properties specific to the bond mode.
slaves	`PIF ref set`	*RO/runtime*	The interfaces which are part of this bond
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: Bond

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given Bond.

Signature:

```
void add_to_other_config (session ref session_id, Bond ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create *Overview:*

Create an interface bond

Signature:

```
Bond ref create (session ref session_id, network ref network, PIF ref
    set members, string MAC, bond_mode mode, (string -> string) map
    properties)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	network	Network to add the bonded PIF to
`PIF ref set`	members	PIFs to add to this bond

type	name	description
string	MAC	The MAC address to use on the bond itself. If this parameter is the empty string then the bond will inherit its MAC address from the primary slave.
bond_mode	mode	Bonding mode to use for the new bond
(string -> string)map	properties	Additional configuration parameters specific to the bond mode

Minimum Role: pool-operator

Return Type: `Bond ref`

The reference of the created Bond object

RPC name: destroy *Overview:*

Destroy an interface bond

Signature:

```
void destroy (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
Bond ref	self	Bond to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the Bonds known to the system.

Signature:

```
Bond ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Bond ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Bond references to Bond records for all Bonds known to the system.

Signature:

```
(Bond ref -> Bond record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Bond ref -> Bond record)map`

records of all objects

RPC name: get_auto_update_mac *Overview:*

Get the auto_update_mac field of the given Bond.

Signature:

```
bool get_auto_update_mac (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the Bond instance with the specified UUID.

Signature:

```
Bond ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Bond ref`

reference to the object

RPC name: get_links_up *Overview:*

Get the links_up field of the given Bond.

Signature:

```
int get_links_up (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_master *Overview:*

Get the master field of the given Bond.

Signature:

```
PIF ref get_master (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_mode *Overview:*

Get the mode field of the given Bond.

Signature:

```
bond_mode get_mode (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bond_mode`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given Bond.

Signature:

```
(string -> string) map get_other_config (session ref session_id, Bond
   ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_primary_slave *Overview:*

Get the primary_slave field of the given Bond.

Signature:

```
PIF ref get_primary_slave (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_properties *Overview:*

Get the properties field of the given Bond.

Signature:

```
(string -> string) map get_properties (session ref session_id, Bond ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Bond.

Signature:

```
Bond record get_record (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Bond record`

all fields from the object

RPC name: get_slaves *Overview:*

Get the slaves field of the given Bond.

Signature:

```
PIF ref set get_slaves (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given Bond.

Signature:

```
string get_uuid (session ref session_id, Bond ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given Bond. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, Bond ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_mode *Overview:*

Change the bond mode

Signature:

```
void set_mode (session ref session_id, Bond ref self, bond_mode value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	The bond
`bond_mode`	value	The new bond mode

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given Bond.

Signature:

```
void set_other_config (session ref session_id, Bond ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_property *Overview:*

Set the value of a property of the bond

Signature:

```
void set_property (session ref session_id, Bond ref self, string name,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Bond ref`	self	The bond
`string`	name	The property name
`string`	value	The property value

Minimum Role: pool-operator

Return Type: **`void`**

Class: Certificate

Description

Fields for class: Certificate

Field	Type	Qualifier	Description
fingerprint	`string`	*RO/constructor*	The certificate's SHA256 fingerprint / hash
host	`host ref`	*RO/constructor*	The host where the certificate is installed
name	`string`	*RO/runtime*	The name of the certificate, only present on certificates of type 'ca'
not_after	`datetime`	*RO/constructor*	Date before which the certificate is valid
not_before	`datetime`	*RO/constructor*	Date after which the certificate is valid
type	`certificate_type`	*RO/runtime*	The type of the certificate, either 'ca', 'host'or 'host_internal'
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: Certificate

RPC name: get_all *Overview:*

Return a list of all the Certificates known to the system.

Signature:

```
Certificate ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Certificate ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Certificate references to Certificate records for all Certificates known to the system.

Signature:

```
(Certificate ref -> Certificate record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Certificate ref -> Certificate record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the Certificate instance with the specified UUID.

Signature:

```
Certificate ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Certificate ref`

reference to the object

RPC name: get_fingerprint *Overview:*

Get the fingerprint field of the given Certificate.

Signature:

```
string get_fingerprint (session ref session_id, Certificate ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given Certificate.

Signature:

```
host ref get_host (session ref session_id, Certificate ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_name *Overview:*

Get the name field of the given Certificate.

Signature:

```
string get_name (session ref session_id, Certificate ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_not_after *Overview:*

Get the not_after field of the given Certificate.

Signature:

```
datetime get_not_after (session ref session_id, Certificate ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_not_before *Overview:*

Get the not_before field of the given Certificate.

Signature:

```
datetime get_not_before (session ref session_id, Certificate ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Certificate.

Signature:

```
Certificate record get_record (session ref session_id, Certificate ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Certificate record`

all fields from the object

RPC name: get_type *Overview:*

Get the type field of the given Certificate.

Signature:

```
certificate_type get_type (session ref session_id, Certificate ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `certificate_type`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given Certificate.

Signature:

```
string get_uuid (session ref session_id, Certificate ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Certificate ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: Cluster

Cluster-wide Cluster metadata

Fields for class: Cluster

Field	Type	Qualifier	Description
allowed_operations	`cluster_operation set`	*RO/runtime*	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.
cluster_config	`(string -> string)map`	*RO/constructor*	Contains read-only settings for the cluster, such as timeouts and other options. It can only be set at cluster create time
cluster_hosts	`Cluster_host ref set`	*RO/runtime*	A list of the cluster_host objects associated with the Cluster
cluster_stack	`string`	*RO/constructor*	Simply the string 'corosync'. No other cluster stacks are currently supported
cluster_token	`string`	*RO/constructor*	The secret key used by xapi-clusterd when it talks to itself on other hosts
current_operations	`(string -> cluster_operation)map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
other_config	`(string -> string)map`	*RW*	Additional configuration

Field	Type	Qualifier	Description
pending_forget	`string set`	*RO/runtime*	Internal field used by Host.destroy to store the IP of cluster members marked as permanently dead but not yet removed
pool_auto_join	`bool`	*RO/constructor*	True if automatically joining new pool members to the cluster. This will be `true` in the first release
token_timeout	`float`	*RO/constructor*	The corosync token timeout in seconds
token_timeout_coefficient	`float`	*RO/constructor*	The corosync token timeout coefficient in seconds
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: Cluster

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given Cluster.

Signature:

```
void add_to_other_config (session ref session_id, Cluster ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

type	name	description
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create *Overview:*

Creates a Cluster object and one Cluster_host object as its first member

Signature:

```
Cluster ref create (session ref session_id, PIF ref PIF, string
    cluster_stack, bool pool_auto_join, float token_timeout, float
    token_timeout_coefficient)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	PIF	The PIF to connect the cluster's first cluster_host to
`string`	cluster_stack	simply the string 'corosync'. No other cluster stacks are currently supported
`bool`	pool_auto_join	true if xapi is automatically joining new pool members to the cluster
`float`	token_timeout	Corosync token timeout in seconds
`float`	token_timeout_coefficient	Corosync token timeout coefficient in seconds

Minimum Role: pool-operator

Return Type: `Cluster ref`

the new Cluster

Possible Error Codes: `INVALID_CLUSTER_STACK`, `INVALID_VALUE`, `PIF_ALLOWS_UNPLUG`, `REQUIRED_PIF_IS_UNPLUGGED`

RPC name: destroy *Overview:*

Destroys a Cluster object and the one remaining Cluster_host member

Signature:

```
void destroy (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	the Cluster to destroy

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `CLUSTER_DOES_NOT_HAVE_ONE_NODE`, `CLUSTER_STACK_IN_USE`

RPC name: get_all *Overview:*

Return a list of all the Clusters known to the system.

Signature:

```
Cluster ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Cluster ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Cluster references to Cluster records for all Clusters known to the system.

Signature:

```
(Cluster ref -> Cluster record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Cluster ref -> Cluster record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given Cluster.

Signature:

```
cluster_operation set get_allowed_operations (session ref session_id,
    Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `cluster_operation set`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the Cluster instance with the specified UUID.

Signature:

```
Cluster ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Cluster ref`

reference to the object

RPC name: get_cluster_config *Overview:*

Get the cluster_config field of the given Cluster.

Signature:

```
(string -> string) map get_cluster_config (session ref session_id,
    Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_cluster_hosts *Overview:*

Get the cluster_hosts field of the given Cluster.

Signature:

```
Cluster_host ref set get_cluster_hosts (session ref session_id, Cluster
     ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Cluster_host ref set`

value of the field

RPC name: get_cluster_stack *Overview:*

Get the cluster_stack field of the given Cluster.

Signature:

```
string get_cluster_stack (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_cluster_token *Overview:*

Get the cluster_token field of the given Cluster.

Signature:

```
string get_cluster_token (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given Cluster.

Signature:

```
(string -> cluster_operation) map get_current_operations (session ref
    session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> cluster_operation)map`

value of the field

RPC name: get_network *Overview:*

Returns the network used by the cluster for inter-host communication, i.e. the network shared by all cluster host PIFs

Signature:

```
network ref get_network (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	the Cluster with the network

Minimum Role: read-only

Return Type: `network ref`

network of cluster

RPC name: get_other_config *Overview:*

Get the other_config field of the given Cluster.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_pending_forget *Overview:*

Get the pending_forget field of the given Cluster.

Signature:

```
string set get_pending_forget (session ref session_id, Cluster ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_pool_auto_join *Overview:*

Get the pool_auto_join field of the given Cluster.

Signature:

```
bool get_pool_auto_join (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Cluster.

Signature:

```
Cluster record get_record (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Cluster record`

all fields from the object

RPC name: get_token_timeout *Overview:*

Get the token_timeout field of the given Cluster.

Signature:

```
float get_token_timeout (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`float`**

value of the field

RPC name: get_token_timeout_coefficient *Overview:*

Get the token_timeout_coefficient field of the given Cluster.

Signature:

```
float get_token_timeout_coefficient (session ref session_id, Cluster
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `float`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given Cluster.

Signature:

```
string get_uuid (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: pool_create *Overview:*

Attempt to create a Cluster from the entire pool

Signature:

```
Cluster ref pool_create (session ref session_id, network ref network,
    string cluster_stack, float token_timeout, float
    token_timeout_coefficient)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	network	the single network on which corosync carries out its inter-host communications
`string`	cluster_stack	simply the string 'corosync'. No other cluster stacks are currently supported
`float`	token_timeout	Corosync token timeout in seconds
`float`	token_timeout_coefficient	Corosync token timeout coefficient in seconds

Minimum Role: pool-operator

Return Type: `Cluster ref`

the new Cluster

RPC name: pool_destroy *Overview:*

Attempt to destroy the Cluster_host objects for all hosts in the pool and then destroy the Cluster.

Signature:

```
void pool_destroy (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	The cluster to destroy.

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `CLUSTER_STACK_IN_USE`, `CLUSTERING_DISABLED`, `CLUSTER_HOST_IS_LAST`

RPC name: pool_force_destroy *Overview:*

Attempt to force destroy the Cluster_host objects, and then destroy the Cluster.

Signature:

```
void pool_force_destroy (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	The cluster to force destroy.

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `CLUSTER_FORCE_DESTROY_FAILED`

RPC name: pool_resync *Overview:*

Resynchronise the cluster_host objects across the pool. Creates them where they need creating and then plugs them

Signature:

```
void pool_resync (session ref session_id, Cluster ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	The cluster to resync

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given Cluster. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, Cluster ref self
    , string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given Cluster.

Signature:

```
void set_other_config (session ref session_id, Cluster ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: Cluster_host

Cluster member metadata

Fields for class: Cluster_host

Field	Type	Qualifier	Description
allowed_operations	`cluster_host_operation set`	*RO/runtime*	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.
cluster	`Cluster ref`	*RO/constructor*	Reference to the Cluster object
current_operations	`(string -> cluster_host_operation)map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
enabled	`bool`	*RO/constructor*	Whether the cluster host believes that clustering should be enabled on this host
host	`host ref`	*RO/constructor*	Reference to the Host object
joined	`bool`	*RO/constructor*	Whether the cluster host has joined the cluster
other_config	`(string -> string)map`	*RO/constructor*	Additional configuration
PIF	`PIF ref`	*RO/constructor*	Reference to the PIF object

Field	Type	Qualifier	Description
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: Cluster_host

RPC name: create *Overview:*

Add a new host to an existing cluster.

Signature:

```
Cluster_host ref create (session ref session_id, Cluster ref cluster,
    host ref host, PIF ref pif)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster ref`	cluster	Cluster to join
`host ref`	host	new cluster member
`PIF ref`	pif	Network interface to use for communication

Minimum Role: pool-operator

Return Type: `Cluster_host ref`

the newly created cluster_host object

Possible Error Codes: `PIF_NOT_ATTACHED_TO_HOST`, `NO_CLUSTER_HOSTS_REACHABLE`

RPC name: destroy *Overview:*

Remove a host from an existing cluster.

Signature:

```
void destroy (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	the cluster_host to remove from the cluster

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `CLUSTER_STACK_IN_USE`, `CLUSTERING_DISABLED`, `CLUSTER_HOST_IS_LAST`

RPC name: disable *Overview:*

Disable cluster membership for an enabled cluster host.

Signature:

```
void disable (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	the cluster_host to disable

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `CLUSTER_STACK_IN_USE`

RPC name: enable *Overview:*

Enable cluster membership for a disabled cluster host.

Signature:

```
void enable (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
Cluster_host ref	self	the cluster_host to enable

Minimum Role: pool-operator

Return Type: **void**

Possible Error Codes: PIF_ALLOWS_UNPLUG, REQUIRED_PIF_IS_UNPLUGGED

RPC name: force_destroy *Overview:*

Remove a host from an existing cluster forcefully.

Signature:

```
void force_destroy (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
Cluster_host ref	self	the cluster_host to remove from the cluster

Minimum Role: pool-operator

Return Type: **void**

Possible Error Codes: CLUSTER_STACK_IN_USE

RPC name: get_all *Overview:*

Return a list of all the Cluster_hosts known to the system.

Signature:

```
Cluster_host ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Cluster_host ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Cluster_host references to Cluster_host records for all Cluster_hosts known to the system.

Signature:

```
(Cluster_host ref -> Cluster_host record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Cluster_host ref -> Cluster_host record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given Cluster_host.

Signature:

```
cluster_host_operation set get_allowed_operations (session ref
    session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `cluster_host_operation set`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the Cluster_host instance with the specified UUID.

Signature:

```
Cluster_host ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Cluster_host ref`

reference to the object

RPC name: get_cluster *Overview:*

Get the cluster field of the given Cluster_host.

Signature:

```
Cluster ref get_cluster (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Cluster ref`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given Cluster_host.

Signature:

```
(string -> cluster_host_operation) map get_current_operations (session
    ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> cluster_host_operation)map`

value of the field

RPC name: get_enabled *Overview:*

Get the enabled field of the given Cluster_host.

Signature:

```
bool get_enabled (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given Cluster_host.

Signature:

```
host ref get_host (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_joined *Overview:*

Get the joined field of the given Cluster_host.

Signature:

```
bool get_joined (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given Cluster_host.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PIF *Overview:*

Get the PIF field of the given Cluster_host.

Signature:

```
PIF ref get_PIF (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Cluster_host.

Signature:

```
Cluster_host record get_record (session ref session_id, Cluster_host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Cluster_host record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given Cluster_host.

Signature:

```
string get_uuid (session ref session_id, Cluster_host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Cluster_host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: console

A console

Fields for class: console

Field	Type	Qualifier	Description
location	string	*RO/runtime*	URI for the console service
other_config	(string -> string)map	*RW*	additional configuration
protocol	console_protocol	*RO/runtime*	the protocol used by this console
uuid	string	*RO/runtime*	Unique identifier/object reference
VM	VM ref	*RO/runtime*	VM to which this console is attached

RPCs associated with class: console

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given console.

Signature:

```
void add_to_other_config (session ref session_id, console ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
console ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: create *Overview:*

Create a new console instance, and return its handle.

Signature:

```
console ref create (session ref session_id, console record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console record`	args	All constructor arguments

Minimum Role: vm-admin

Return Type: `console ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified console instance.

Signature:

```
void destroy (session ref session_id, console ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: vm-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the consoles known to the system.

Signature:

```
console ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `console ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of console references to console records for all consoles known to the system.

Signature:

```
(console ref -> console record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(console ref -> console record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the console instance with the specified UUID.

Signature:

```
console ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `console ref`

reference to the object

RPC name: get_location *Overview:*

Get the location field of the given console.

Signature:

```
string get_location (session ref session_id, console ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given console.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    console ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_protocol *Overview:*

Get the protocol field of the given console.

Signature:

```
console_protocol get_protocol (session ref session_id, console ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: read-only

Return Type: `console_protocol`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given console.

Signature:

```
console record get_record (session ref session_id, console ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: read-only

Return Type: `console record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given console.

Signature:

```
string get_uuid (session ref session_id, console ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VM *Overview:*

Get the VM field of the given console.

Signature:

```
VM ref get_VM (session ref session_id, console ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given console. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, console ref self
    , string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given console.

Signature:

```
void set_other_config (session ref session_id, console ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`console ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

Class: crashdump

This class is deprecated.

A VM crashdump

Fields for class: crashdump

Field	Type	Qualifier	Description
other_config	`(string -> string)map`	*RW*	**Deprecated**. additional configuration
uuid	`string`	*RO/runtime*	**Deprecated**. Unique identifier/object reference
VDI	`VDI ref`	*RO/constructor*	**Deprecated**. the virtual disk
VM	`VM ref`	*RO/constructor*	**Deprecated**. the virtual machine

RPCs associated with class: crashdump

RPC name: add_to_other_config **This message is deprecated.**

Overview:

Add the given key-value pair to the other_config field of the given crashdump.

Signature:

```
void add_to_other_config (session ref session_id, crashdump ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object

type	name	description
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: destroy   This message is deprecated.

Overview:

Destroy the specified crashdump

Signature:

```
void destroy (session ref session_id, crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	The crashdump to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all   This message is deprecated.

Overview:

Return a list of all the crashdumps known to the system.

Signature:

```
crashdump ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `crashdump ref set`

references to all objects

RPC name: get_all_records   This message is deprecated.

Overview:

Return a map of crashdump references to crashdump records for all crashdumps known to the system.

Signature:

```
(crashdump ref -> crashdump record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(crashdump ref -> crashdump record)map`

records of all objects

RPC name: get_by_uuid   This message is deprecated.

Overview:

Get a reference to the crashdump instance with the specified UUID.

Signature:

```
crashdump ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `crashdump ref`

reference to the object

RPC name: get_other_config   This message is deprecated.

Overview:

Get the other_config field of the given crashdump.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record This message is deprecated.

Overview:

Get a record containing the current state of the given crashdump.

Signature:

```
crashdump record get_record (session ref session_id, crashdump ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `crashdump record`

all fields from the object

RPC name: get_uuid   This message is deprecated.

Overview:

Get the uuid field of the given crashdump.

Signature:

```
string get_uuid (session ref session_id, crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VDI   This message is deprecated.

Overview:

Get the VDI field of the given crashdump.

Signature:

```
VDI ref get_VDI (session ref session_id, crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

RPC name: get_VM This message is deprecated.

Overview:

Get the VM field of the given crashdump.

Signature:

```
VM ref get_VM (session ref session_id, crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: remove_from_other_config This message is deprecated.

Overview:

Remove the given key and its corresponding value from the other_config field of the given crashdump. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, crashdump ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config **This message is deprecated.**

Overview:

Set the other_config field of the given crashdump.

Signature:

```
void set_other_config (session ref session_id, crashdump ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`crashdump ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Class: data_source

Data sources for logging in RRDs

Fields for class: data_source

Field	Type	Qualifier	Description
enabled	`bool`	*RO/runtime*	true if the data source is being logged
max	**`float`**	*RO/runtime*	the maximum value of the data source
min	**`float`**	*RO/runtime*	the minimum value of the data source
name_description	`string`	*RO/runtime*	a notes field containing human-readable description

Field	Type	Qualifier	Description
name_label	`string`	*RO/runtime*	a human-readable name
standard	`bool`	*RO/runtime*	true if the data source is enabled by default. Non-default data sources cannot be disabled
units	`string`	*RO/runtime*	the units of the value
value	**`float`**	*RO/runtime*	current value of the data source

RPCs associated with class: data_source

Class data_source has no additional RPCs associated with it.

Class: DR_task

DR task

Fields for class: DR_task

Field	Type	Qualifier	Description
introduced_SRs	`SR ref set`	*RO/runtime*	All SRs introduced by this appliance
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: DR_task

RPC name: create *Overview:*

Create a disaster recovery task which will query the supplied list of devices

Signature:

```
DR_task ref create (session ref session_id, string type, (string ->
    string) map device_config, string set whitelist)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	type	The SR driver type of the SRs to introduce
`(string -> string)map`	device_config	The device configuration of the SRs to introduce
`string set`	whitelist	The devices to use for disaster recovery

Minimum Role: pool-operator

Return Type: `DR_task ref`

The reference to the created task

RPC name: destroy *Overview:*

Destroy the disaster recovery task, detaching and forgetting any SRs introduced which are no longer required

Signature:

```
void destroy (session ref session_id, DR_task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`DR_task ref`	self	The disaster recovery task to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the DR_tasks known to the system.

Signature:

```
DR_task ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `DR_task ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of DR_task references to DR_task records for all DR_tasks known to the system.

Signature:

```
(DR_task ref -> DR_task record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(DR_task ref -> DR_task record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the DR_task instance with the specified UUID.

Signature:

```
DR_task ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `DR_task ref`

reference to the object

RPC name: get_introduced_SRs *Overview:*

Get the introduced_SRs field of the given DR_task.

Signature:

```
SR ref set get_introduced_SRs (session ref session_id, DR_task ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`DR_task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given DR_task.

Signature:

```
DR_task record get_record (session ref session_id, DR_task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`DR_task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `DR_task record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given DR_task.

Signature:

```
string get_uuid (session ref session_id, DR_task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`DR_task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: event

Asynchronous event registration and handling

Fields for class: event

Field	Type	Qualifier	Description
class	`string`	*RO/constructor*	The name of the class of the object that changed
id	**`int`**	*RO/constructor*	An ID, monotonically increasing, and local to the current session
obj_uuid	`string`	*RO/constructor*	**Deprecated**. The uuid of the object that changed
operation	`event_operation`	*RO/constructor*	The operation that was performed

Field	Type	Qualifier	Description
ref	string	*RO/constructor*	A reference to the object that changed
timestamp	datetime	*RO/constructor*	**Deprecated**. The time at which the event occurred
snapshot	<object record>	*RO/runtime*	The record of the database object that was added, changed or deleted

RPCs associated with class: event

RPC name: from *Overview:*

Blocking call which returns a new token and a (possibly empty) batch of events. The returned token can be used in subsequent calls to this function.

Signature:

```
<event batch> from (session ref session_id, string set classes, string
    token, float timeout)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
string set	classes	register for events for the indicated classes
string	token	A token representing the point from which to generate database events. The empty string represents the beginning.
float	timeout	Return after this many seconds if no events match

Minimum Role: read-only

Return Type: an event batch

a structure consisting of a token ('token'), a map of valid references per object type ('valid_ref_counts'), and a set of event records ('events').

Possible Error Codes: `SESSION_NOT_REGISTERED`, `EVENTS_LOST`

RPC name: get_current_id *Overview:*

Return the ID of the next event to be generated by the system

Signature:

```
int get_current_id (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `int`

the event ID

RPC name: inject *Overview:*

Injects an artificial event on the given object and returns the corresponding ID in the form of a token, which can be used as a point of reference for database events. For example, to check whether an object has reached the right state before attempting an operation, one can inject an artificial event on the object and wait until the token returned by consecutive event.from calls is lexicographically greater than the one returned by event.inject.

Signature:

```
string inject (session ref session_id, string class, string ref)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	class	class of the object
`string`	ref	A reference to the object that will be changed.

Minimum Role: read-only

Return Type: `string`

the event ID in the form of a token

RPC name: next This message is deprecated.

Overview:

Blocking call which returns a (possibly empty) batch of events. This method is only recommended for legacy use. New development should use event.from which supercedes this method.

Signature:

```
event record set next (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `event record set`

A set of events

Possible Error Codes: `SESSION_NOT_REGISTERED`, `EVENTS_LOST`

RPC name: register This message is deprecated.

Overview:

Registers this session with the event system for a set of given classes. This method is only recommended for legacy use in conjunction with event.next.

Signature:

```
void register (session ref session_id, string set classes)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string set`	classes	the classes for which the session will register with the event system; specifying * as the desired class will register for all classes

Minimum Role: read-only

Return Type: `void`

RPC name: unregister **This message is deprecated.**

Overview:

Removes this session's registration with the event system for a set of given classes. This method is only recommended for legacy use in conjunction with event.next.

Signature:

```
void unregister (session ref session_id, string set classes)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string set`	classes	the classes for which the session's registration with the event system will be removed

Minimum Role: read-only

Return Type: `void`

Class: Feature

A new piece of functionality

Fields for class: Feature

Field	Type	Qualifier	Description
enabled	`bool`	*RO/runtime*	Indicates whether the feature is enabled
experimental	`bool`	*RO/constructor*	Indicates whether the feature is experimental (as opposed to stable and fully supported)

Field	Type	Qualifier	Description
host	`host ref`	*RO/runtime*	The host where this feature is available
name_description	`string`	*RO/constructor*	a notes field containing human-readable description
name_label	`string`	*RO/constructor*	a human-readable name
uuid	`string`	*RO/runtime*	Unique identifier/object reference
version	`string`	*RO/constructor*	The version of this feature

RPCs associated with class: Feature

RPC name: get_all *Overview:*

Return a list of all the Features known to the system.

Signature:

```
Feature ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Feature ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Feature references to Feature records for all Features known to the system.

Signature:

```
(Feature ref -> Feature record) map get_all_records (session ref
   session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Feature ref -> Feature record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the Feature instances with the given label.

Signature:

```
Feature ref set get_by_name_label (session ref session_id, string label
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `Feature ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the Feature instance with the specified UUID.

Signature:

```
Feature ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Feature ref`

reference to the object

RPC name: get_enabled *Overview:*

Get the enabled field of the given Feature.

Signature:

```
bool get_enabled (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_experimental *Overview:*

Get the experimental field of the given Feature.

Signature:

```
bool get_experimental (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given Feature.

Signature:

```
host ref get_host (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given Feature.

Signature:

```
string get_name_description (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given Feature.

Signature:

```
string get_name_label (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Feature.

Signature:

```
Feature record get_record (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Feature record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given Feature.

Signature:

```
string get_uuid (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_version *Overview:*

Get the version field of the given Feature.

Signature:

```
string get_version (session ref session_id, Feature ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Feature ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: GPU_group

A group of compatible GPUs across the resource pool

Fields for class: GPU_group

Field	Type	Qualifier	Description
allocation_algorithm	`allocation_algorithm`	*RW*	Current allocation of vGPUs to pGPUs for this group
enabled_VGPU_types	`VGPU_type ref set`	*RO/runtime*	vGPU types supported on at least one of the pGPUs in this group
GPU_types	`string set`	*RO/runtime*	List of GPU types (vendor+device ID) that can be in this group
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
other_config	`(string -> string)map`	*RW*	Additional configuration
PGPUs	`PGPU ref set`	*RO/runtime*	List of pGPUs in the group
supported_VGPU_types	`VGPU_type ref set`	*RO/runtime*	vGPU types supported on at least one of the pGPUs in this group
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VGPUs	`VGPU ref set`	*RO/runtime*	List of vGPUs using the group

RPCs associated with class: GPU_group

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given GPU_group.

Signature:

```
void add_to_other_config (session ref session_id, GPU_group ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create *Overview:*

Signature:

```
GPU_group ref create (session ref session_id, string name_label, string
     name_description, (string -> string) map other_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name_label	
`string`	name_description	
`(string -> string)map`	other_config	

Minimum Role: pool-operator

Return Type: `GPU_group ref`

RPC name: destroy *Overview:*

Signature:

```
void destroy (session ref session_id, GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	The GPU group to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the GPU_groups known to the system.

Signature:

```
GPU_group ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `GPU_group ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of GPU_group references to GPU_group records for all GPU_groups known to the system.

Signature:

```
(GPU_group ref -> GPU_group record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(GPU_group ref -> GPU_group record)map`

records of all objects

RPC name: get_allocation_algorithm *Overview:*

Get the allocation_algorithm field of the given GPU_group.

Signature:

```
allocation_algorithm get_allocation_algorithm (session ref session_id,
    GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `allocation_algorithm`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the GPU_group instances with the given label.

Signature:

```
GPU_group ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `GPU_group ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the GPU_group instance with the specified UUID.

Signature:

```
GPU_group ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `GPU_group ref`

reference to the object

RPC name: get_enabled_VGPU_types *Overview:*

Get the enabled_VGPU_types field of the given GPU_group.

Signature:

```
VGPU_type ref set get_enabled_VGPU_types (session ref session_id,
    GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type ref set`

value of the field

RPC name: get_GPU_types *Overview:*

Get the GPU_types field of the given GPU_group.

Signature:

```
string set get_GPU_types (session ref session_id, GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given GPU_group.

Signature:

```
string get_name_description (session ref session_id, GPU_group ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given GPU_group.

Signature:

```
string get_name_label (session ref session_id, GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given GPU_group.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PGPUs *Overview:*

Get the PGPUs field of the given GPU_group.

Signature:

```
PGPU ref set get_PGPUs (session ref session_id, GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
GPU_group ref	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU ref set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given GPU_group.

Signature:

```
GPU_group record get_record (session ref session_id, GPU_group ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
GPU_group ref	self	reference to the object

Minimum Role: read-only

Return Type: `GPU_group record`

all fields from the object

RPC name: get_remaining_capacity *Overview:*

Signature:

```
int get_remaining_capacity (session ref session_id, GPU_group ref self,
    VGPU_type ref vgpu_type)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	The GPU group to query
`VGPU_type ref`	vgpu_type	The VGPU_type for which the remaining capacity will be calculated

Minimum Role: read-only

Return Type: `int`

The number of VGPUs of the given type which can still be started on the PGPUs in the group

RPC name: get_supported_VGPU_types *Overview:*

Get the supported_VGPU_types field of the given GPU_group.

Signature:

```
VGPU_type ref set get_supported_VGPU_types (session ref session_id,
    GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given GPU_group.

Signature:

```
string get_uuid (session ref session_id, GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VGPUs *Overview:*

Get the VGPUs field of the given GPU_group.

Signature:

```
VGPU ref set get_VGPUs (session ref session_id, GPU_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU ref set`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given GPU_group. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, GPU_group ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_allocation_algorithm *Overview:*

Set the allocation_algorithm field of the given GPU_group.

Signature:

```
void set_allocation_algorithm (session ref session_id, GPU_group ref
    self, allocation_algorithm value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object
`allocation_algorithm`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_description *Overview:*

Set the name/description field of the given GPU_group.

Signature:

```
void set_name_description (session ref session_id, GPU_group ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name/label field of the given GPU_group.

Signature:

```
void set_name_label (session ref session_id, GPU_group ref self, string
     value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given GPU_group.

Signature:

```
void set_other_config (session ref session_id, GPU_group ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`GPU_group ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Class: host

A physical host

Fields for class: host

Field	Type	Qualifier	Description
address	`string`	*RW*	The address by which this host can be contacted from any other host in the pool

Field	Type	Qualifier	Description
allowed_operations	`host_allowed_operations set`	*RO/runtime*	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.
API_version_major	`int`	*RO/runtime*	major version number
API_version_minor	`int`	*RO/runtime*	minor version number
API_version_vendor	`string`	*RO/runtime*	identification of vendor
API_version_vendor_implementation	`(string -> string)map`	*RO/runtime*	details of vendor implementation
bios_strings	`(string -> string)map`	*RO/runtime*	BIOS strings
blobs	`(string -> blob ref)map`	*RO/runtime*	Binary blobs associated with this host
capabilities	`string set`	*RO/constructor*	Xen capabilities
certificates	`Certificate ref set`	*RO/runtime*	List of certificates installed in the host
chipset_info	`(string -> string)map`	*RO/runtime*	Information about chipset features
control_domain	`VM ref`	*RO/runtime*	The control domain (domain 0)
cpu_configuration	`(string -> string)map`	*RO/runtime*	The CPU configuration on this host. May contain keys such as “nr_nodes”, “sockets_per_node”, “cores_per_socket”, or “threads_per_core”
cpu_info	`(string -> string)map`	*RO/runtime*	Details about the physical CPUs on this host

Field	Type	Qualifier	Description
crash_dump_sr	`SR ref`	*RW*	The SR in which VDIs for crash dumps are created
crashdumps	`host_crashdump ref set`	*RO/runtime*	Set of host crash dumps
current_operations	`(string -> host_allowed_operations )map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
display	`host_display`	*RW*	indicates whether the host is configured to output its console to a physical display device
edition	`string`	*RO/runtime*	Product edition
editions	`string set`	*RO/runtime*	List of all available product editions
enabled	`bool`	*RO/runtime*	True if the host is currently enabled
external_auth_configuration	`(string -> string)map`	*RO/runtime*	configuration specific to external authentication service
external_auth_service_name	`string`	*RO/runtime*	name of external authentication service configured; empty if none configured.
external_auth_type	`string`	*RO/runtime*	type of external authentication service configured; empty if none configured.
features	`Feature ref set`	*RO/runtime*	List of features available on this host
guest_VCPUs_params	`(string -> string)map`	*RW*	VCPUs params to apply to all resident guests

Field	Type	Qualifier	Description
ha_network_peers	`string set`	*RO/runtime*	The set of hosts visible via the network from this host
ha_statefiles	`string set`	*RO/runtime*	The set of statefiles accessible from this host
host_CPUs	`host_cpu ref set`	*RO/runtime*	The physical CPUs on this host
hostname	`string`	*RW*	The hostname of this host
https_only	`bool`	*RO/runtime*	Reflects whether port 80 is open (false) or not (true)
iscsi_iqn	`string`	*RO/constructor*	The initiator IQN for the host
last_software_update	`datetime`	*RO/runtime*	Date and time when the last software update was applied
latest_synced_updates_applied	`latest_synced_updates_applied_state`	*RO/runtime*	Default as 'unknown', 'yes'if the host is up to date with updates synced from remote CDN, otherwise 'no'
license_params	`(string -> string)map`	*RO/runtime*	State of the current license
license_server	`(string -> string)map`	*RW*	Contact information of the license server
local_cache_sr	`SR ref`	*RO/constructor*	The SR that is used as a local cache
logging	`(string -> string)map`	*RW*	logging configuration
memory_overhead	`int`	*RO/runtime*	Virtualization memory overhead (bytes).
metrics	`host_metrics ref`	*RO/runtime*	metrics associated with this host

Field	Type	Qualifier	Description
multipathing	`bool`	*RO/constructor*	Specifies whether multipathing is enabled
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
numa_affinity_policy	`host_numa_affinity_policy`	*RO/runtime*	NUMA-aware VM memory and vCPU placement policy
other_config	`(string -> string)map`	*RW*	additional configuration
patches	`host_patch ref set`	*RO/runtime*	**Deprecated**. Set of host patches
PBDs	`PBD ref set`	*RO/runtime*	physical blockdevices
PCIs	`PCI ref set`	*RO/runtime*	List of PCI devices in the host
pending_guidances	`update_guidances set`	*RO/runtime*	The set of pending guidances after applying updates
PGPUs	`PGPU ref set`	*RO/runtime*	List of physical GPUs in the host
PIFs	`PIF ref set`	*RO/runtime*	physical network interfaces
power_on_config	`(string -> string)map`	*RO/runtime*	The power on config
power_on_mode	`string`	*RO/runtime*	The power on mode
PUSBs	`PUSB ref set`	*RO/runtime*	List of physical USBs in the host
resident_VMs	`VM ref set`	*RO/runtime*	list of VMs currently resident on host
sched_policy	`string`	*RO/runtime*	Scheduler policy currently in force on this host

Field	Type	Qualifier	Description
software_version	`(string -> string)map`	*RO/constructor*	version strings
ssl_legacy	`bool`	*RO/constructor*	**Deprecated**. Allow SSLv3 protocol and ciphersuites as used by older server versions. This controls both incoming and outgoing connections. When this is set to a different value, the host immediately restarts its SSL/TLS listening service; typically this takes less than a second but existing connections to it will be broken. API login sessions will remain valid.
supported_bootloaders	`string set`	*RO/runtime*	a list of the bootloaders installed on the machine
suspend_image_sr	`SR ref`	*RW*	The SR in which VDIs for suspend images are created
tags	`string set`	*RW*	user-specified tags for categorization purposes
tls_verification_enabled	`bool`	*RO/runtime*	True if this host has TLS verifcation enabled
uefi_certificates	`string`	*RO/constructor*	**Deprecated**. The UEFI certificates allowing Secure Boot
updates	`pool_update ref set`	*RO/runtime*	Set of updates

Field	Type	Qualifier	Description
updates_requiring_reboot	pool_update ref set	*RO/runtime*	List of updates which require reboot
uuid	string	*RO/runtime*	Unique identifier/object reference
virtual_hardware_platform_versions	int set	*RO/runtime*	The set of versions of the virtual hardware platform that the host can offer to its guests

RPCs associated with class: host

RPC name: add_tags *Overview:*

Add the given value to the tags field of the given host. If the value is already in that Set, then do nothing.

Signature:

```
void add_tags (session ref session_id, host ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	self	reference to the object
string	value	New value to add

Minimum Role: vm-operator

Return Type: **void**

RPC name: add_to_guest_VCPUs_params *Overview:*

Add the given key-value pair to the guest_VCPUs_params field of the given host.

Signature:

```
void add_to_guest_VCPUs_params (session ref session_id, host ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: add_to_license_server *Overview:*

Add the given key-value pair to the license_server field of the given host.

Signature:

```
void add_to_license_server (session ref session_id, host ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: add_to_logging *Overview:*

Add the given key-value pair to the logging field of the given host.

Signature:

```
void add_to_logging (session ref session_id, host ref self, string key,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given host.

Signature:

```
void add_to_other_config (session ref session_id, host ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: apply_edition *Overview:*

Change to another edition, or reactivate the current edition after a license has expired. This may be subject to the successful checkout of an appropriate license.

Signature:

```
void apply_edition (session ref session_id, host ref host, string
    edition, bool force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	edition	The requested edition
`bool`	force	Update the license params even if the apply call fails

Minimum Role: pool-operator

Return Type: `void`

RPC name: apply_recommended_guidances **This message is removed.**

Overview:

apply all recommended guidances both on the host and on all HVM VMs on the host after updates are applied on the host

Signature:

```
void apply_recommended_guidances (session ref session_id, host ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host whose recommended guidances will be applied

Minimum Role: pool-operator

Return Type: `void`

RPC name: apply_updates *Overview:*

apply updates from current enabled repository on a host

Signature:

```
string set set apply_updates (session ref session_id, host ref self,
    string hash)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host where updates will be applied
`string`	hash	The hash of updateinfo to be applied which is returned by previous pool.sync_udpates

Minimum Role: client-cert

Return Type: `string set set`

The list of results after applying updates, including livepatch apply failures and recommended guidances

RPC name: assert_can_evacuate *Overview:*

Check this host can be evacuated.

Signature:

```
void assert_can_evacuate (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host to evacuate

Minimum Role: pool-operator

Return Type: **void**

RPC name: backup_rrds *Overview:*

This causes the RRDs to be backed up to the master

Signature:

```
void backup_rrds (session ref session_id, host ref host, float delay)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	Schedule a backup of the RRDs of this host
float	delay	Delay in seconds from when the call is received to perform the backup

Minimum Role: pool-admin

Return Type: **void**

RPC name: bugreport_upload *Overview:*

Run xen-bugtool --yestoall and upload the output to support

Signature:

```
void bugreport_upload (session ref session_id, host ref host, string
    url, (string -> string) map options)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host on which to run xen-bugtool
string	url	The URL to upload to
(string -> string)map	options	Extra configuration operations

Minimum Role: pool-operator

Return Type: **void**

RPC name: call_extension *Overview:*

Call an API extension on this host

Signature:

```
string call_extension (session ref session_id, host ref host, string
    call)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host
string	call	Rpc call for the extension

Minimum Role: pool-admin

Return Type: string

Result from the extension

RPC name: call_plugin *Overview:*

Call an API plugin on this host

Signature:

```
string call_plugin (session ref session_id, host ref host, string
    plugin, string fn, (string -> string) map args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	plugin	The name of the plugin
`string`	fn	The name of the function within the plugin
`(string -> string)map`	args	Arguments for the function

Minimum Role: pool-admin

Return Type: `string`

Result from the plugin

RPC name: compute_free_memory *Overview:*

Computes the amount of free memory on the host.

Signature:

```
int compute_free_memory (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to send the request to

Minimum Role: read-only

Return Type: `int`

the amount of free memory on the host.

RPC name: compute_memory_overhead *Overview:*

Computes the virtualization memory overhead of a host.

Signature:

```
int compute_memory_overhead (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host for which to compute the memory overhead

Minimum Role: read-only

Return Type: `int`

the virtualization memory overhead of the host.

RPC name: create_new_blob *Overview:*

Create a placeholder for a named binary blob of data that is associated with this host

Signature:

```
blob ref create_new_blob (session ref session_id, host ref host, string
     name, string mime_type, bool public)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	name	The name associated with the blob
`string`	mime_type	The mime type for the data. Empty string translates to application/octet-stream

type	name	description
`bool`	public	True if the blob should be publicly available

Minimum Role: pool-operator

Return Type: `blob ref`

The reference of the blob, needed for populating its data

RPC name: declare_dead *Overview:*

Declare that a host is dead. This is a dangerous operation, and should only be called if the administrator is absolutely sure the host is definitely dead

Signature:

```
void declare_dead (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to declare is dead

Minimum Role: pool-operator

Return Type: `void`

RPC name: destroy *Overview:*

Destroy specified host record in database

Signature:

```
void destroy (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	self	The host record to remove

Minimum Role: pool-operator

Return Type: **void**

RPC name: disable *Overview:*

Puts the host into a state in which no new VMs can be started. Currently active VMs on the host continue to execute.

Signature:

```
void disable (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The Host to disable

Minimum Role: client-cert

Return Type: **void**

RPC name: disable_display *Overview:*

Disable console output to the physical display device next time this host boots

Signature:

```
host_display disable_display (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host

Minimum Role: pool-operator

Return Type: `host_display`

This host's physical display usage

RPC name: disable_external_auth *Overview:*

This call disables external authentication on the local host

Signature:

```
void disable_external_auth (session ref session_id, host ref host, (
    string -> string) map config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host whose external authentication should be disabled
(string -> string)map	config	Optional parameters as a list of key-values containing the configuration data

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: disable_local_storage_caching *Overview:*

Disable the use of a local SR for caching purposes

Signature:

```
void disable_local_storage_caching (session ref session_id, host ref
    host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host

Minimum Role: pool-operator

Return Type: `void`

RPC name: dmesg *Overview:*

Get the host xen dmesg.

Signature:

```
string dmesg (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to query

Minimum Role: pool-operator

Return Type: `string`

dmesg string

RPC name: dmesg_clear *Overview:*

Get the host xen dmesg, and clear the buffer.

Signature:

```
string dmesg_clear (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to query

Minimum Role: pool-operator

Return Type: `string`

dmesg string

RPC name: emergency_disable_tls_verification *Overview:*

Disable TLS verification for this host only

Signature:

```
void emergency_disable_tls_verification (session ref session_id)
<!--NeedCopy-->
```

Return Type: **`void`**

RPC name: emergency_ha_disable *Overview:*

This call disables HA on the local host. This should only be used with extreme care.

Signature:

```
void emergency_ha_disable (session ref session_id, bool soft)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`bool`	soft	Disable HA temporarily, revert upon host reboot or further changes, idempotent

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: emergency_reenable_tls_verification *Overview:*

Reenable TLS verification for this host only

Signature:

```
void emergency_reenable_tls_verification (session ref session_id)
<!--NeedCopy-->
```

Return Type: `void`

RPC name: emergency_reset_server_certificate *Overview:*

Delete the current TLS server certificate and replace by a new, self-signed one. This should only be used with extreme care.

Signature:

```
void emergency_reset_server_certificate (session ref session_id)
<!--NeedCopy-->
```

Return Type: `void`

RPC name: enable *Overview:*

Puts the host into a state in which new VMs can be started.

Signature:

```
void enable (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to enable

Minimum Role: client-cert

Return Type: `void`

RPC name: enable_display *Overview:*

Enable console output to the physical display device next time this host boots

Signature:

```
host_display enable_display (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host

Minimum Role: pool-operator

Return Type: `host_display`

This host's physical display usage

RPC name: enable_external_auth *Overview:*

This call enables external authentication on a host

Signature:

```
void enable_external_auth (session ref session_id, host ref host, (
    string -> string) map config, string service_name, string auth_type)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host whose external authentication should be enabled
`(string -> string)map`	config	A list of key-values containing the configuration data
`string`	service_name	The name of the service
`string`	auth_type	The type of authentication (e.g. AD for Active Directory)

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: enable_local_storage_caching *Overview:*

Enable the use of a local SR for caching purposes

Signature:

```
void enable_local_storage_caching (session ref session_id, host ref
    host, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`SR ref`	sr	The SR to use as a local cache

Minimum Role: pool-operator

Return Type: `void`

RPC name: evacuate *Overview:*

Migrate all VMs off of this host, where possible.

Signature:

```
void evacuate (session ref session_id, host ref host, network ref
    network, int evacuate_batch_size)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to evacuate
`network ref`	network	Optional preferred network for migration

type	name	description
`int`	evacuate_batch_size	The maximum number of VMs to be migrated per batch 0 will use the value `evacuation&#45;batch&#45;size` defined in xapi.conf

Minimum Role: client-cert

Return Type: `void`

RPC name: forget_data_source_archives *Overview:*

Forget the recorded statistics related to the specified data source

Signature:

```
void forget_data_source_archives (session ref session_id, host ref host
    , string data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	data_source	The data source whose archives are to be forgotten

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_address *Overview:*

Get the address field of the given host.

Signature:

```
string get_address (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_all *Overview:*

Return a list of all the hosts known to the system.

Signature:

```
host ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `host ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of host references to host records for all hosts known to the system.

Signature:

```
(host ref -> host record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(host ref -> host record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given host.

Signature:

```
host_allowed_operations set get_allowed_operations (session ref
    session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_allowed_operations set`

value of the field

RPC name: get_API_version_major *Overview:*

Get the API_version/major field of the given host.

Signature:

```
int get_API_version_major (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_API_version_minor *Overview:*

Get the API_version/minor field of the given host.

Signature:

```
int get_API_version_minor (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_API_version_vendor *Overview:*

Get the API_version/vendor field of the given host.

Signature:

```
string get_API_version_vendor (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_API_version_vendor_implementation *Overview:*

Get the API_version/vendor_implementation field of the given host.

Signature:

```
(string -> string) map get_API_version_vendor_implementation (session
    ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_bios_strings *Overview:*

Get the bios_strings field of the given host.

Signature:

```
(string -> string) map get_bios_strings (session ref session_id, host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_blobs *Overview:*

Get the blobs field of the given host.

Signature:

```
(string -> blob ref) map get_blobs (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> blob ref)map`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the host instances with the given label.

Signature:

```
host ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `host ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the host instance with the specified UUID.

Signature:

```
host ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `host ref`

reference to the object

RPC name: get_capabilities *Overview:*

Get the capabilities field of the given host.

Signature:

```
string set get_capabilities (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_certificates *Overview:*

Get the certificates field of the given host.

Signature:

```
Certificate ref set get_certificates (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Certificate ref set`

value of the field

RPC name: get_chipset_info *Overview:*

Get the chipset_info field of the given host.

Signature:

```
(string -> string) map get_chipset_info (session ref session_id, host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_control_domain *Overview:*

Get the control_domain field of the given host.

Signature:

```
VM ref get_control_domain (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: get_cpu_configuration *Overview:*

Get the cpu_configuration field of the given host.

Signature:

```
(string -> string) map get_cpu_configuration (session ref session_id,
    host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_cpu_info *Overview:*

Get the cpu_info field of the given host.

Signature:

```
(string -> string) map get_cpu_info (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_crash_dump_sr *Overview:*

Get the crash_dump_sr field of the given host.

Signature:

```
SR ref get_crash_dump_sr (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_crashdumps *Overview:*

Get the crashdumps field of the given host.

Signature:

```
host_crashdump ref set get_crashdumps (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_crashdump ref set`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given host.

Signature:

```
(string -> host_allowed_operations) map get_current_operations (session
    ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> host_allowed_operations)map`

value of the field

RPC name: get_data_sources *Overview:*

Signature:

```
data_source record set get_data_sources (session ref session_id, host
    ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to interrogate

Minimum Role: read-only

Return Type: `data_source record set`

A set of data sources

RPC name: get_display *Overview:*

Get the display field of the given host.

Signature:

```
host_display get_display (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_display`

value of the field

RPC name: get_edition *Overview:*

Get the edition field of the given host.

Signature:

```
string get_edition (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_editions *Overview:*

Get the editions field of the given host.

Signature:

```
string set get_editions (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_enabled *Overview:*

Get the enabled field of the given host.

Signature:

```
bool get_enabled (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_external_auth_configuration *Overview:*

Get the external_auth_configuration field of the given host.

Signature:

```
(string -> string) map get_external_auth_configuration (session ref
    session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_external_auth_service_name *Overview:*

Get the external_auth_service_name field of the given host.

Signature:

```
string get_external_auth_service_name (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_external_auth_type *Overview:*

Get the external_auth_type field of the given host.

Signature:

```
string get_external_auth_type (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_features *Overview:*

Get the features field of the given host.

Signature:

```
Feature ref set get_features (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Feature ref set`

value of the field

RPC name: get_guest_VCPUs_params *Overview:*

Get the guest_VCPUs_params field of the given host.

Signature:

```
(string -> string) map get_guest_VCPUs_params (session ref session_id,
    host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_ha_network_peers *Overview:*

Get the ha_network_peers field of the given host.

Signature:

```
string set get_ha_network_peers (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_ha_statefiles *Overview:*

Get the ha_statefiles field of the given host.

Signature:

```
string set get_ha_statefiles (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_host_CPUs *Overview:*

Get the host_CPUs field of the given host.

Signature:

```
host_cpu ref set get_host_CPUs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_cpu ref set`

value of the field

RPC name: get_hostname *Overview:*

Get the hostname field of the given host.

Signature:

```
string get_hostname (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_https_only *Overview:*

Get the https_only field of the given host.

Signature:

```
bool get_https_only (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_iscsi_iqn *Overview:*

Get the iscsi_iqn field of the given host.

Signature:

```
string get_iscsi_iqn (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_last_software_update *Overview:*

Get the last_software_update field of the given host.

Signature:

```
datetime get_last_software_update (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_latest_synced_updates_applied *Overview:*

Get the latest_synced_updates_applied field of the given host.

Signature:

```
latest_synced_updates_applied_state get_latest_synced_updates_applied (
    session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `latest_synced_updates_applied_state`

value of the field

RPC name: get_license_params *Overview:*

Get the license_params field of the given host.

Signature:

```
(string -> string) map get_license_params (session ref session_id, host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_license_server *Overview:*

Get the license_server field of the given host.

Signature:

```
(string -> string) map get_license_server (session ref session_id, host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_local_cache_sr *Overview:*

Get the local_cache_sr field of the given host.

Signature:

```
SR ref get_local_cache_sr (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_log *Overview:*

Get the host's log file

Signature:

```
string get_log (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to query

Minimum Role: read-only

Return Type: `string`

The contents of the host's primary log file

RPC name: get_logging *Overview:*

Get the logging field of the given host.

Signature:

```
(string -> string) map get_logging (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_management_interface *Overview:*

Returns the management interface for the specified host

Signature:

```
PIF ref get_management_interface (session ref session_id, host ref host
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	Which host's management interface is required

Minimum Role: pool-operator

Return Type: `PIF ref`

The management interface for the host

RPC name: get_memory_overhead *Overview:*

Get the memory/overhead field of the given host.

Signature:

```
int get_memory_overhead (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_metrics *Overview:*

Get the metrics field of the given host.

Signature:

```
host_metrics ref get_metrics (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_metrics ref`

value of the field

RPC name: get_multipathing *Overview:*

Get the multipathing field of the given host.

Signature:

```
bool get_multipathing (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given host.

Signature:

```
string get_name_description (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given host.

Signature:

```
string get_name_label (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_numa_affinity_policy *Overview:*

Get the numa_affinity_policy field of the given host.

Signature:

```
host_numa_affinity_policy get_numa_affinity_policy (session ref
    session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_numa_affinity_policy`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given host.

Signature:

```
(string -> string) map get_other_config (session ref session_id, host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_patches **This message is deprecated.**

Overview:

Get the patches field of the given host.

Signature:

```
host_patch ref set get_patches (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_patch ref set`

value of the field

RPC name: get_PBDs *Overview:*

Get the PBDs field of the given host.

Signature:

```
PBD ref set get_PBDs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PBD ref set`

value of the field

RPC name: get_PCIs *Overview:*

Get the PCIs field of the given host.

Signature:

```
PCI ref set get_PCIs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI ref set`

value of the field

RPC name: get_pending_guidances *Overview:*

Get the pending_guidances field of the given host.

Signature:

```
update_guidances set get_pending_guidances (session ref session_id,
    host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `update_guidances set`

value of the field

RPC name: get_PGPUs *Overview:*

Get the PGPUs field of the given host.

Signature:

```
PGPU ref set get_PGPUs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU ref set`

value of the field

RPC name: get_PIFs *Overview:*

Get the PIFs field of the given host.

Signature:

```
PIF ref set get_PIFs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref set`

value of the field

RPC name: get_power_on_config *Overview:*

Get the power_on_config field of the given host.

Signature:

```
(string -> string) map get_power_on_config (session ref session_id,
    host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_power_on_mode *Overview:*

Get the power_on_mode field of the given host.

Signature:

```
string get_power_on_mode (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_PUSBs *Overview:*

Get the PUSBs field of the given host.

Signature:

```
PUSB ref set get_PUSBs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PUSB ref set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given host.

Signature:

```
host record get_record (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host record`

all fields from the object

RPC name: get_resident_VMs *Overview:*

Get the resident_VMs field of the given host.

Signature:

```
VM ref set get_resident_VMs (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref set`

value of the field

RPC name: get_sched_gran *Overview:*

Gets xen's sched-gran on a host

Signature:

```
host_sched_gran get_sched_gran (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host

Return Type: `host_sched_gran`

The host's sched-gran

RPC name: get_sched_policy *Overview:*

Get the sched_policy field of the given host.

Signature:

```
string get_sched_policy (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_server_certificate *Overview:*

Get the installed server public TLS certificate.

Signature:

```
string get_server_certificate (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host

Minimum Role: read-only

Return Type: `string`

The installed server public TLS certificate, in PEM form.

RPC name: get_server_localtime *Overview:*

This call queries the host's clock for the current time in the host's local timezone

Signature:

```
datetime get_server_localtime (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host whose clock should be queried

Minimum Role: read-only

Return Type: `datetime`

The current local time

RPC name: get_servertime *Overview:*

This call queries the host's clock for the current time

Signature:

```
datetime get_servertime (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host whose clock should be queried

Minimum Role: read-only

Return Type: `datetime`

The current time

RPC name: get_software_version *Overview:*

Get the software_version field of the given host.

Signature:

```
(string -> string) map get_software_version (session ref session_id,
    host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_ssl_legacy **This message is deprecated.**

Overview:

Get the ssl_legacy field of the given host.

Signature:

```
bool get_ssl_legacy (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_supported_bootloaders *Overview:*

Get the supported_bootloaders field of the given host.

Signature:

```
string set get_supported_bootloaders (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_suspend_image_sr *Overview:*

Get the suspend_image_sr field of the given host.

Signature:

```
SR ref get_suspend_image_sr (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_system_status_capabilities *Overview:*

Signature:

```
string get_system_status_capabilities (session ref session_id, host ref
    host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to interrogate

Minimum Role: read-only

Return Type: `string`

An XML fragment containing the system status capabilities.

RPC name: get_tags *Overview:*

Get the tags field of the given host.

Signature:

```
string set get_tags (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_tls_verification_enabled *Overview:*

Get the tls_verification_enabled field of the given host.

Signature:

```
bool get_tls_verification_enabled (session ref session_id, host ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_uefi_certificates **This message is deprecated.**

Overview:

Get the uefi_certificates field of the given host.

Signature:

```
string get_uefi_certificates (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uncooperative_resident_VMs **This message is deprecated.**

Overview:

Return a set of VMs which are not co-operating with the host's memory control system

Signature:

```
VM ref set get_uncooperative_resident_VMs (session ref session_id, host
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host to query

Minimum Role: read-only

Return Type: `VM ref set`

VMs which are not co-operating

RPC name: get_updates *Overview:*

Get the updates field of the given host.

Signature:

```
pool_update ref set get_updates (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_update ref set`

value of the field

RPC name: get_updates_requiring_reboot *Overview:*

Get the updates_requiring_reboot field of the given host.

Signature:

```
pool_update ref set get_updates_requiring_reboot (session ref
    session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_update ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given host.

Signature:

```
string get_uuid (session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_virtual_hardware_platform_versions *Overview:*

Get the virtual_hardware_platform_versions field of the given host.

Signature:

```
int set get_virtual_hardware_platform_versions (session ref session_id,
    host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int set`

value of the field

RPC name: get_vms_which_prevent_evacuation *Overview:*

Return a set of VMs which prevent the host being evacuated, with per-VM error codes

Signature:

```
(VM ref -> string set) map get_vms_which_prevent_evacuation (session
    ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host to query

Minimum Role: read-only

Return Type: `(VM ref -> string set)map`

VMs which block evacuation together with reasons

RPC name: has_extension *Overview:*

Return true if the extension is available on the host

Signature:

```
bool has_extension (session ref session_id, host ref host, string name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	name	The name of the API call

Minimum Role: pool-admin

Return Type: `bool`

True if the extension exists, false otherwise

RPC name: install_server_certificate *Overview:*

Install the TLS server certificate.

Signature:

```
void install_server_certificate (session ref session_id, host ref host,
    string certificate, string private_key, string certificate_chain)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	certificate	The server certificate, in PEM form
`string`	private_key	The unencrypted private key used to sign the certificate, in PKCS#8 form
`string`	certificate_chain	The certificate chain, in PEM form

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: license_add *Overview:*

Apply a new license to a host

Signature:

```
void license_add (session ref session_id, host ref host, string
    contents)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to upload the license to
`string`	contents	The contents of the license file, base64 encoded

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `LICENSE_PROCESSING_ERROR`

RPC name: license_apply **This message is removed.**

Overview:

Apply a new license to a host

Signature:

```
void license_apply (session ref session_id, host ref host, string
    contents)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to upload the license to
`string`	contents	The contents of the license file, base64 encoded

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `LICENSE_PROCESSING_ERROR`

RPC name: license_remove *Overview:*

Remove any license file from the specified host, and switch that host to the unlicensed edition

Signature:

```
void license_remove (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host from which any license will be removed

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: list_methods *Overview:*

List all supported methods

Signature:

```
string set list_methods (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `string set`

The name of every supported method.

RPC name: local_management_reconfigure *Overview:*

Reconfigure the management network interface. Should only be used if Host.management_reconfigure is impossible because the network configuration is broken.

Signature:

```
void local_management_reconfigure (session ref session_id, string
    interface)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	interface	name of the interface to use as a management interface

Minimum Role: pool-operator

Return Type: `void`

RPC name: management_disable *Overview:*

Disable the management network interface

Signature:

```
void management_disable (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: management_reconfigure *Overview:*

Reconfigure the management network interface

Signature:

```
void management_reconfigure (session ref session_id, PIF ref pif)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	pif	reference to a PIF object corresponding to the management interface

Minimum Role: pool-operator

Return Type: `void`

RPC name: migrate_receive *Overview:*

Prepare to receive a VM, returning a token which can be passed to VM.migrate.

Signature:

```
(string -> string) map migrate_receive (session ref session_id, host
    ref host, network ref network, (string -> string) map options)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The target host
`network ref`	network	The network through which migration traffic should be received.
`(string -> string)map`	options	Extra configuration operations

Minimum Role: vm-power-admin

Return Type: `(string -> string)map`

A value which should be passed to VM.migrate

RPC name: power_on *Overview:*

Attempt to power-on the host (if the capability exists).

Signature:

```
void power_on (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to power on

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: query_data_source *Overview:*

Query the latest value of the specified data source

Signature:

```
float query_data_source (session ref session_id, host ref host, string
    data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	data_source	The data source to query

Minimum Role: read-only

Return Type: **`float`**

The latest value, averaged over the last 5 seconds

RPC name: reboot *Overview:*

Reboot the host. (This function can only be called if there are no currently running VMs on the host and it is disabled.)

Signature:

```
void reboot (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to reboot

Minimum Role: pool-operator

Return Type: `void`

RPC name: record_data_source *Overview:*

Start recording the specified data source

Signature:

```
void record_data_source (session ref session_id, host ref host, string
    data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	data_source	The data source to record

Minimum Role: pool-operator

Return Type: `void`

RPC name: refresh_pack_info **This message is deprecated.**

Overview:

Refresh the list of installed Supplemental Packs.

Signature:

```
void refresh_pack_info (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to modify

Minimum Role: pool-operator

Return Type: `void`

RPC name: refresh_server_certificate *Overview:*

Replace the internal self-signed host certficate with a new one.

Signature:

```
void refresh_server_certificate (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host

Minimum Role: pool-admin

Return Type: `void`

RPC name: remove_from_guest_VCPUs_params *Overview:*

Remove the given key and its corresponding value from the guest_VCPUs_params field of the given host. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_guest_VCPUs_params (session ref session_id, host ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_from_license_server *Overview:*

Remove the given key and its corresponding value from the license_server field of the given host. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_license_server (session ref session_id, host ref self,
     string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_from_logging *Overview:*

Remove the given key and its corresponding value from the logging field of the given host. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_logging (session ref session_id, host ref self, string
    key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given host. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, host ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_tags *Overview:*

Remove the given value from the tags field of the given host. If the value is not in that Set, then do nothing.

Signature:

```
void remove_tags (session ref session_id, host ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	self	reference to the object
string	value	Value to remove

Minimum Role: vm-operator

Return Type: `void`

RPC name: reset_cpu_features **This message is removed.**

Overview:

Remove the feature mask, such that after a reboot all features of the CPU are enabled.

Signature:

```
void reset_cpu_features (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host

Minimum Role: pool-operator

Return Type: `void`

RPC name: reset_server_certificate *Overview:*

Delete the current TLS server certificate and replace by a new, self-signed one. This should only be used with extreme care.

Signature:

```
void reset_server_certificate (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host

Minimum Role: pool-admin

Return Type: `void`

RPC name: restart_agent *Overview:*

Restarts the agent after a 10 second pause. WARNING: this is a dangerous operation. Any operations in progress will be aborted, and unrecoverable data loss may occur. The caller is responsible for ensuring that there are no operations in progress when this method is called.

Signature:

```
void restart_agent (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host on which you want to restart the agent

Minimum Role: pool-operator

Return Type: `void`

RPC name: retrieve_wlb_evacuate_recommendations *Overview:*

Retrieves recommended host migrations to perform when evacuating the host from the wlb server. If a VM cannot be migrated from the host the reason is listed instead of a recommendation.

Signature:

```
(VM ref -> string set) map retrieve_wlb_evacuate_recommendations (
    session ref session_id, host ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host to query

Minimum Role: read-only

Return Type: `(VM ref -> string set)map`

VMs and the reasons why they would block evacuation, or their target host recommended by the wlb server

RPC name: send_debug_keys *Overview:*

Inject the given string as debugging keys into Xen

Signature:

```
void send_debug_keys (session ref session_id, host ref host, string
    keys)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	keys	The keys to send

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_address *Overview:*

Set the address field of the given host.

Signature:

```
void set_address (session ref session_id, host ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_cpu_features **This message is removed.**

Overview:

Set the CPU features to be used after a reboot, if the given features string is valid.

Signature:

```
void set_cpu_features (session ref session_id, host ref host, string
    features)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	features	The features string (32 hexadecimal digits)

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_crash_dump_sr *Overview:*

Set the crash_dump_sr field of the given host.

Signature:

```
void set_crash_dump_sr (session ref session_id, host ref self, SR ref
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`SR ref`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_display *Overview:*

Set the display field of the given host.

Signature:

```
void set_display (session ref session_id, host ref self, host_display
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`host_display`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_guest_VCPUs_params *Overview:*

Set the guest_VCPUs_params field of the given host.

Signature:

```
void set_guest_VCPUs_params (session ref session_id, host ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_hostname *Overview:*

Set the hostname field of the given host.

Signature:

```
void set_hostname (session ref session_id, host ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_hostname_live *Overview:*

Sets the host name to the specified string. Both the API and lower-level system hostname are changed immediately.

Signature:

```
void set_hostname_live (session ref session_id, host ref host, string
    hostname)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host whose host name to set
`string`	hostname	The new host name

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `HOST_NAME_INVALID`

RPC name: set_https_only *Overview:*

updates the host firewall to open or close port 80 depending on the value

Signature:

```
void set_https_only (session ref session_id, host ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The Host
`bool`	value	true - http port 80 will be blocked, false - http port 80 will be open

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_iscsi_iqn *Overview:*

Sets the initiator IQN for the host

Signature:

```
void set_iscsi_iqn (session ref session_id, host ref host, string value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	value	The value to which the IQN should be set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_license_server *Overview:*

Set the license_server field of the given host.

Signature:

```
void set_license_server (session ref session_id, host ref self, (string
     -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_logging *Overview:*

Set the logging field of the given host.

Signature:

```
void set_logging (session ref session_id, host ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_multipathing *Overview:*

Specifies whether multipathing is enabled

Signature:

```
void set_multipathing (session ref session_id, host ref host, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host
bool	value	Whether multipathing should be enabled

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_name_description *Overview:*

Set the name/description field of the given host.

Signature:

```
void set_name_description (session ref session_id, host ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_name_label *Overview:*

Set the name/label field of the given host.

Signature:

```
void set_name_label (session ref session_id, host ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_numa_affinity_policy *Overview:*

Set VM placement NUMA affinity policy

Signature:

```
void set_numa_affinity_policy (session ref session_id, host ref self,
    host_numa_affinity_policy value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	self	The host
host_numa_affinity_policy	value	The NUMA affinity policy to apply to a host

Minimum Role: pool-admin

Return Type: **void**

RPC name: set_other_config *Overview:*

Set the other_config field of the given host.

Signature:

```
void set_other_config (session ref session_id, host ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_power_on_mode *Overview:*

Set the power-on-mode, host, user and password

Signature:

```
void set_power_on_mode (session ref session_id, host ref self, string
    power_on_mode, (string -> string) map power_on_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host
`string`	power_on_mode	power-on-mode can be empty, wake-on-lan, DRAC or other
`(string -> string)map`	power_on_config	Power on config

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_sched_gran *Overview:*

Sets xen's sched-gran on a host. See: https://xenbits.xen.org/docs/unstable/misc/xen-command-line.html#sched-gran-x86

Signature:

```
void set_sched_gran (session ref session_id, host ref self,
    host_sched_gran value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`host ref`	self	The host
`host_sched_gran`	value	The sched-gran to apply to a host

Return Type: **`void`**

RPC name: set_ssl_legacy *Overview:*

Enable/disable SSLv3 for interoperability with older server versions. When this is set to a different value, the host immediately restarts its SSL/TLS listening service; typically this takes less than a second but existing connections to it will be broken. API login sessions will remain valid.

Signature:

```
void set_ssl_legacy (session ref session_id, host ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	The host
`bool`	value	True to allow SSLv3 and ciphersuites as used in old XenServer versions

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_suspend_image_sr *Overview:*

Set the suspend_image_sr field of the given host.

Signature:

```
void set_suspend_image_sr (session ref session_id, host ref self, SR
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`SR ref`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_tags *Overview:*

Set the tags field of the given host.

Signature:

```
void set_tags (session ref session_id, host ref self, string set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	self	reference to the object
`string set`	value	New value to set

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: set_uefi_certificates **This message is deprecated.**

Overview:

Sets the UEFI certificates on a host

Signature:

```
void set_uefi_certificates (session ref session_id, host ref host,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host
`string`	value	The certificates to apply to a host

Return Type: **`void`**

RPC name: shutdown *Overview:*

Shutdown the host. (This function can only be called if there are no currently running VMs on the host and it is disabled.)

Signature:

```
void shutdown (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The Host to shutdown

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: shutdown_agent *Overview:*

Shuts the agent down after a 10 second pause. WARNING: this is a dangerous operation. Any operations in progress will be aborted, and unrecoverable data loss may occur. The caller is responsible for ensuring that there are no operations in progress when this method is called.

Signature:

```
void shutdown_agent (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: sync_data *Overview:*

This causes the synchronisation of the non-database data (messages, RRDs and so on) stored on the master to be synchronised with the host

Signature:

```
void sync_data (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to whom the data should be sent

Minimum Role: pool-admin

Return Type: `void`

RPC name: syslog_reconfigure *Overview:*

Re-configure syslog logging

Signature:

```
void syslog_reconfigure (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	Tell the host to reread its Host.logging parameters and reconfigure itself accordingly

Minimum Role: pool-operator

Return Type: `void`

Class: host_cpu

This class is deprecated.

A physical CPU

Fields for class: host_cpu

Field	Type	Qualifier	Description
family	`int`	*RO/runtime*	**Deprecated**. the family (number) of the physical CPU
features	`string`	*RO/runtime*	**Deprecated**. the physical CPU feature bitmap
flags	`string`	*RO/runtime*	**Deprecated**. the flags of the physical CPU (a decoded version of the features field)
host	`host ref`	*RO/runtime*	**Deprecated**. the host the CPU is in
model	`int`	*RO/runtime*	**Deprecated**. the model number of the physical CPU
modelname	`string`	*RO/runtime*	**Deprecated**. the model name of the physical CPU
number	`int`	*RO/runtime*	**Deprecated**. the number of the physical CPU within the host
other_config	`(string -> string)map`	*RW*	**Deprecated**. additional configuration
speed	`int`	*RO/runtime*	**Deprecated**. the speed of the physical CPU

Field	Type	Qualifier	Description
stepping	`string`	*RO/runtime*	**Deprecated**. the stepping of the physical CPU
utilisation	`float`	*RO/runtime*	**Deprecated**. the current CPU utilisation
uuid	`string`	*RO/runtime*	**Deprecated**. Unique identifier/object reference
vendor	`string`	*RO/runtime*	**Deprecated**. the vendor of the physical CPU

RPCs associated with class: host_cpu

RPC name: add_to_other_config   This message is deprecated.

Overview:

Add the given key-value pair to the other_config field of the given host_cpu.

Signature:

```
void add_to_other_config (session ref session_id, host_cpu ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all   This message is deprecated.

Overview:

Return a list of all the host_cpus known to the system.

Signature:

```
host_cpu ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `host_cpu ref set`

references to all objects

RPC name: get_all_records   This message is deprecated.

Overview:

Return a map of host_cpu references to host_cpu records for all host_cpus known to the system.

Signature:

```
(host_cpu ref -> host_cpu record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(host_cpu ref -> host_cpu record)map`

records of all objects

RPC name: get_by_uuid   This message is deprecated.

Overview:

Get a reference to the host_cpu instance with the specified UUID.

Signature:

```
host_cpu ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `host_cpu ref`

reference to the object

RPC name: get_family **This message is deprecated.**

Overview:

Get the family field of the given host_cpu.

Signature:

```
int get_family (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_features **This message is deprecated.**

Overview:

Get the features field of the given host_cpu.

Signature:

```
string get_features (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_flags **This message is deprecated.**

Overview:

Get the flags field of the given host_cpu.

Signature:

```
string get_flags (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_host **This message is deprecated.**

Overview:

Get the host field of the given host_cpu.

Signature:

```
host ref get_host (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_model This message is deprecated.

Overview:

Get the model field of the given host_cpu.

Signature:

```
int get_model (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_modelname This message is deprecated.

Overview:

Get the modelname field of the given host_cpu.

Signature:

```
string get_modelname (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_number This message is deprecated.

Overview:

Get the number field of the given host_cpu.

Signature:

```
int get_number (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_other_config **This message is deprecated.**

Overview:

Get the other_config field of the given host_cpu.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record **This message is deprecated.**

Overview:

Get a record containing the current state of the given host_cpu.

Signature:

```
host_cpu record get_record (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_cpu record`

all fields from the object

RPC name: get_speed **This message is deprecated.**

Overview:

Get the speed field of the given host_cpu.

Signature:

```
int get_speed (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_stepping **This message is deprecated.**

Overview:

Get the stepping field of the given host_cpu.

Signature:

```
string get_stepping (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_utilisation This message is deprecated.

Overview:

Get the utilisation field of the given host_cpu.

Signature:

```
float get_utilisation (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `float`

value of the field

RPC name: get_uuid This message is deprecated.

Overview:

Get the uuid field of the given host_cpu.

Signature:

```
string get_uuid (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor This message is deprecated.

Overview:

Get the vendor field of the given host_cpu.

Signature:

```
string get_vendor (session ref session_id, host_cpu ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host_cpu ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config This message is deprecated.

Overview:

Remove the given key and its corresponding value from the other_config field of the given host_cpu. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, host_cpu ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host_cpu ref	self	reference to the object
string	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config **This message is deprecated.**

Overview:

Set the other_config field of the given host_cpu.

Signature:

```
void set_other_config (session ref session_id, host_cpu ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_cpu ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: host_crashdump

Represents a host crash dump

Fields for class: host_crashdump

Field	Type	Qualifier	Description
host	`host ref`	*RO/constructor*	Host the crashdump relates to
other_config	`(string -> string)map`	*RW*	additional configuration
size	`int`	*RO/runtime*	Size of the crashdump
timestamp	`datetime`	*RO/runtime*	Time the crash happened

Field	Type	Qualifier	Description
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: host_crashdump

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given host_crashdump.

Signature:

```
void add_to_other_config (session ref session_id, host_crashdump ref
    self, string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: destroy *Overview:*

Destroy specified host crash dump, removing it from the disk.

Signature:

```
void destroy (session ref session_id, host_crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host_crashdump ref	self	The host crashdump to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the host_crashdumps known to the system.

Signature:

```
host_crashdump ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `host_crashdump ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of host_crashdump references to host_crashdump records for all host_crashdumps known to the system.

Signature:

```
(host_crashdump ref -> host_crashdump record) map get_all_records (
    session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(host_crashdump ref -> host_crashdump record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the host_crashdump instance with the specified UUID.

Signature:

```
host_crashdump ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `host_crashdump ref`

reference to the object

RPC name: get_host *Overview:*

Get the host field of the given host_crashdump.

Signature:

```
host ref get_host (session ref session_id, host_crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given host_crashdump.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    host_crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given host_crashdump.

Signature:

```
host_crashdump record get_record (session ref session_id,
    host_crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_crashdump record`

all fields from the object

RPC name: get_size *Overview:*

Get the size field of the given host_crashdump.

Signature:

```
int get_size (session ref session_id, host_crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_timestamp *Overview:*

Get the timestamp field of the given host_crashdump.

Signature:

```
datetime get_timestamp (session ref session_id, host_crashdump ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given host_crashdump.

Signature:

```
string get_uuid (session ref session_id, host_crashdump ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given host_crashdump. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, host_crashdump
    ref self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given host_crashdump.

Signature:

```
void set_other_config (session ref session_id, host_crashdump ref self,
    (string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: upload *Overview:*

Upload the specified host crash dump to a specified URL

Signature:

```
void upload (session ref session_id, host_crashdump ref self, string
    url, (string -> string) map options)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_crashdump ref`	self	The host crashdump to upload
`string`	url	The URL to upload to
`(string -> string)map`	options	Extra configuration operations

Minimum Role: pool-operator

Return Type: `void`

Class: host_metrics

The metrics associated with a host

Fields for class: host_metrics

Field	Type	Qualifier	Description
last_updated	`datetime`	*RO/runtime*	Time at which this information was last updated
live	`bool`	*RO/runtime*	Pool master thinks this host is live
memory_free	`int`	*RO/runtime*	**Removed**. Free host memory (bytes)
memory_total	`int`	*RO/runtime*	Total host memory (bytes)
other_config	`(string -> string)map`	*RW*	additional configuration
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: host_metrics

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given host_metrics.

Signature:

```
void add_to_other_config (session ref session_id, host_metrics ref self
    , string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the host_metrics instances known to the system.

Signature:

```
host_metrics ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `host_metrics ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of host_metrics references to host_metrics records for all host_metrics instances known to the system.

Signature:

```
(host_metrics ref -> host_metrics record) map get_all_records (session
   ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(host_metrics ref -> host_metrics record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the host_metrics instance with the specified UUID.

Signature:

```
host_metrics ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `host_metrics ref`

reference to the object

RPC name: get_last_updated *Overview:*

Get the last_updated field of the given host_metrics.

Signature:

```
datetime get_last_updated (session ref session_id, host_metrics ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_live *Overview:*

Get the live field of the given host_metrics.

Signature:

```
bool get_live (session ref session_id, host_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_memory_free **This message is removed.**

Overview:

Get the memory/free field of the given host_metrics.

Signature:

```
int get_memory_free (session ref session_id, host_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_memory_total *Overview:*

Get the memory/total field of the given host_metrics.

Signature:

```
int get_memory_total (session ref session_id, host_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`int`**

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given host_metrics.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    host_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given host_metrics.

Signature:

```
host_metrics record get_record (session ref session_id, host_metrics
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_metrics record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given host_metrics.

Signature:

```
string get_uuid (session ref session_id, host_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given host_metrics. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, host_metrics ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given host_metrics.

Signature:

```
void set_other_config (session ref session_id, host_metrics ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_metrics ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: host_patch

This class is deprecated.

Represents a patch stored on a server

Fields for class: host_patch

Field	Type	Qualifier	Description
applied	`bool`	*RO/runtime*	**Deprecated**. True if the patch has been applied
host	`host ref`	*RO/constructor*	**Deprecated**. Host the patch relates to
name_description	`string`	*RO/constructor*	**Deprecated**. a notes field containing human-readable description
name_label	`string`	*RO/constructor*	**Deprecated**. a human-readable name
other_config	`(string -> string) map`	*RW*	**Deprecated**. additional configuration
pool_patch	`pool_patch ref`	*RO/constructor*	**Deprecated**. The patch applied
size	`int`	*RO/runtime*	**Deprecated**. Size of the patch
timestamp_applied	`datetime`	*RO/runtime*	**Deprecated**. Time the patch was applied
uuid	`string`	*RO/runtime*	**Deprecated**. Unique identifier/object reference
version	`string`	*RO/constructor*	**Deprecated**. Patch version number

RPCs associated with class: host_patch

RPC name: add_to_other_config **This message is deprecated.**

Overview:

Add the given key-value pair to the other_config field of the given host_patch.

Signature:

```
void add_to_other_config (session ref session_id, host_patch ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: apply This message is deprecated.

Overview:

Apply the selected patch and return its output

Signature:

```
string apply (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	The patch to apply

Minimum Role: pool-operator

Return Type: `string`

the output of the patch application process

RPC name: destroy   This message is deprecated.

Overview:

Destroy the specified host patch, removing it from the disk. This does NOT reverse the patch

Signature:

```
void destroy (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	The patch to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all   This message is deprecated.

Overview:

Return a list of all the host_patchs known to the system.

Signature:

```
host_patch ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `host_patch ref set`

references to all objects

RPC name: get_all_records   This message is deprecated.

Overview:

Return a map of host_patch references to host_patch records for all host_patchs known to the system.

Signature:

```
(host_patch ref -> host_patch record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(host_patch ref -> host_patch record)map`

records of all objects

RPC name: get_applied This message is deprecated.

Overview:

Get the applied field of the given host_patch.

Signature:

```
bool get_applied (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_by_name_label This message is deprecated.

Overview:

Get all the host_patch instances with the given label.

Signature:

```
host_patch ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `host_patch ref set`

references to objects with matching names

RPC name: get_by_uuid This message is deprecated.

Overview:

Get a reference to the host_patch instance with the specified UUID.

Signature:

```
host_patch ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `host_patch ref`

reference to the object

RPC name: get_host This message is deprecated.

Overview:

Get the host field of the given host_patch.

Signature:

```
host ref get_host (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_name_description This message is deprecated.

Overview:

Get the name/description field of the given host_patch.

Signature:

```
string get_name_description (session ref session_id, host_patch ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label This message is deprecated.

Overview:

Get the name/label field of the given host_patch.

Signature:

```
string get_name_label (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config This message is deprecated.

Overview:

Get the other_config field of the given host_patch.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_pool_patch **This message is deprecated.**

Overview:

Get the pool_patch field of the given host_patch.

Signature:

```
pool_patch ref get_pool_patch (session ref session_id, host_patch ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_patch ref`

value of the field

RPC name: get_record **This message is deprecated.**

Overview:

Get a record containing the current state of the given host_patch.

Signature:

```
host_patch record get_record (session ref session_id, host_patch ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_patch record`

all fields from the object

RPC name: get_size This message is deprecated.

Overview:

Get the size field of the given host_patch.

Signature:

```
int get_size (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host_patch ref	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_timestamp_applied This message is deprecated.

Overview:

Get the timestamp_applied field of the given host_patch.

Signature:

```
datetime get_timestamp_applied (session ref session_id, host_patch ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host_patch ref	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_uuid   This message is deprecated.

Overview:

Get the uuid field of the given host_patch.

Signature:

```
string get_uuid (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_version   This message is deprecated.

Overview:

Get the version field of the given host_patch.

Signature:

```
string get_version (session ref session_id, host_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config This message is deprecated.

Overview:

Remove the given key and its corresponding value from the other_config field of the given host_patch. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, host_patch ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config This message is deprecated.

Overview:

Set the other_config field of the given host_patch.

Signature:

```
void set_other_config (session ref session_id, host_patch ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host_patch ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: LVHD

LVHD SR specific operations

Fields for class: LVHD

Field	Type	Qualifier	Description
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: LVHD

RPC name: enable_thin_provisioning *Overview:*

Upgrades an LVHD SR to enable thin-provisioning. Future VDIs created in this SR will be thinly-provisioned, although existing VDIs will be left alone. Note that the SR must be attached to the SRmaster for upgrade to work.

Signature:

```
string enable_thin_provisioning (session ref session_id, host ref host,
     SR ref SR, int initial_allocation, int allocation_quantum)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The LVHD Host to upgrade to being thin-provisioned.
`SR ref`	SR	The LVHD SR to upgrade to being thin-provisioned.
`int`	initial_allocation	The initial amount of space to allocate to a newly-created VDI in bytes

type	name	description
`int`	allocation_quantum	The amount of space to allocate to a VDI when it needs to be enlarged in bytes

Minimum Role: pool-admin

Return Type: `string`

Message from LVHD.enable_thin_provisioning extension

RPC name: get_by_uuid *Overview:*

Get a reference to the LVHD instance with the specified UUID.

Signature:

```
LVHD ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `LVHD ref`

reference to the object

RPC name: get_record *Overview:*

Get a record containing the current state of the given LVHD.

Signature:

```
LVHD record get_record (session ref session_id, LVHD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`LVHD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `LVHD record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given LVHD.

Signature:

```
string get_uuid (session ref session_id, LVHD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`LVHD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: message

An message for the attention of the administrator

Fields for class: message

Field	Type	Qualifier	Description
body	`string`	*RO/runtime*	The body of the message
cls	`cls`	*RO/runtime*	The class of the object this message is associated with
name	`string`	*RO/runtime*	The name of the message
obj_uuid	`string`	*RO/runtime*	The uuid of the object this message is associated with
priority	`int`	*RO/runtime*	The message priority, 0 being low priority
timestamp	`datetime`	*RO/runtime*	The time at which the message was created
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: message

RPC name: create *Overview:*

Signature:

```
message ref create (session ref session_id, string name, int priority,
    cls cls, string obj_uuid, string body)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	The name of the message
`int`	priority	The priority of the message
`cls`	cls	The class of object this message is associated with

type	name	description
string	obj_uuid	The uuid of the object this message is associated with
string	body	The body of the message

Minimum Role: pool-operator

Return Type: `message ref`

The reference of the created message

RPC name: destroy *Overview:*

Signature:

```
void destroy (session ref session_id, message ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
message ref	self	The reference of the message to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: destroy_many *Overview:*

Signature:

```
void destroy_many (session ref session_id, message ref set messages)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`message ref set`	messages	Messages to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get *Overview:*

Signature:

```
(message ref -> message record) map get (session ref session_id, cls
    cls, string obj_uuid, datetime since)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`cls`	cls	The class of object
`string`	obj_uuid	The uuid of the object
`datetime`	since	The cutoff time

Minimum Role: read-only

Return Type: `(message ref -> message record)map`

The relevant messages

RPC name: get_all *Overview:*

Signature:

```
message ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `message ref set`

The references to the messages

RPC name: get_all_records *Overview:*

Signature:

```
(message ref -> message record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(message ref -> message record)map`

The messages

RPC name: get_all_records_where *Overview:*

Signature:

```
(message ref -> message record) map get_all_records_where (session ref
    session_id, string expr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	expr	The expression to match (not currently used)

Minimum Role: read-only

Return Type: `(message ref -> message record)map`

The messages

RPC name: get_by_uuid *Overview:*

Signature:

```
message ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	The uuid of the message

Minimum Role: read-only

Return Type: `message ref`

The message reference

RPC name: get_record *Overview:*

Signature:

```
message record get_record (session ref session_id, message ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`message ref`	self	The reference to the message

Minimum Role: read-only

Return Type: `message record`

The message record

RPC name: get_since *Overview:*

Signature:

```
(message ref -> message record) map get_since (session ref session_id,
    datetime since)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`datetime`	since	The cutoff time

Minimum Role: read-only

Return Type: `(message ref -> message record)map`

The relevant messages

Class: network

A virtual network

Fields for class: network

Field	Type	Qualifier	Description
allowed_operations	`network_operations set`	*RO/runtime*	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.
assigned_ips	`(VIF ref -> string)map`	*RO/runtime*	The IP addresses assigned to VIFs on networks that have active xapi-managed DHCP
blobs	`(string -> blob ref)map`	*RO/runtime*	Binary blobs associated with this network
bridge	`string`	*RO/constructor*	name of the bridge corresponding to this network on the local host

Field	Type	Qualifier	Description
current_operations	`(string -> network_operations)map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
default_locking_mode	`network_default_locking_mode`	*RO/runtime*	The network will use this value to determine the behaviour of all VIFs where locking_mode = default
managed	`bool`	*RO/constructor*	true if the bridge is managed by xapi
MTU	**`int`**	*RW*	MTU in octets
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
other_config	`(string -> string)map`	*RW*	additional configuration
PIFs	`PIF ref set`	*RO/runtime*	list of connected pifs
purpose	`network_purpose set`	*RO/runtime*	Set of purposes for which the server will use this network
tags	`string set`	*RW*	user-specified tags for categorization purposes
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VIFs	`VIF ref set`	*RO/runtime*	list of connected vifs

RPCs associated with class: network

RPC name: add_purpose *Overview:*

Give a network a new purpose (if not present already)

Signature:

```
void add_purpose (session ref session_id, network ref self,
    network_purpose value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	The network
`network_purpose`	value	The purpose to add

Minimum Role: pool-admin

Return Type: `void`

Possible Error Codes: `NETWORK_INCOMPATIBLE_PURPOSES`

RPC name: add_tags *Overview:*

Add the given value to the tags field of the given network. If the value is already in that Set, then do nothing.

Signature:

```
void add_tags (session ref session_id, network ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`string`	value	New value to add

Minimum Role: vm-operator

Return Type: `void`

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given network.

Signature:

```
void add_to_other_config (session ref session_id, network ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: `void`

RPC name: create *Overview:*

Create a new network instance, and return its handle.

Signature:

```
network ref create (session ref session_id, network record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network record`	args	All constructor arguments

Minimum Role: vm-admin

Return Type: `network ref`

reference to the newly created object

RPC name: create_new_blob *Overview:*

Create a placeholder for a named binary blob of data that is associated with this pool

Signature:

```
blob ref create_new_blob (session ref session_id, network ref network,
    string name, string mime_type, bool public)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	network	The network
`string`	name	The name associated with the blob
`string`	mime_type	The mime type for the data. Empty string translates to application/octet-stream
`bool`	public	True if the blob should be publicly available

Minimum Role: pool-operator

Return Type: `blob ref`

The reference of the blob, needed for populating its data

RPC name: destroy *Overview:*

Destroy the specified network instance.

Signature:

```
void destroy (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: vm-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the networks known to the system.

Signature:

```
network ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `network ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of network references to network records for all networks known to the system.

Signature:

```
(network ref -> network record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(network ref -> network record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given network.

Signature:

```
network_operations set get_allowed_operations (session ref session_id,
    network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network_operations set`

value of the field

RPC name: get_assigned_ips *Overview:*

Get the assigned_ips field of the given network.

Signature:

```
(VIF ref -> string) map get_assigned_ips (session ref session_id,
    network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(VIF ref -> string)map`

value of the field

RPC name: get_blobs *Overview:*

Get the blobs field of the given network.

Signature:

```
(string -> blob ref) map get_blobs (session ref session_id, network ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> blob ref)map`

value of the field

RPC name: get_bridge *Overview:*

Get the bridge field of the given network.

Signature:

```
string get_bridge (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the network instances with the given label.

Signature:

```
network ref set get_by_name_label (session ref session_id, string label
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `network ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the network instance with the specified UUID.

Signature:

```
network ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `network ref`

reference to the object

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given network.

Signature:

```
(string -> network_operations) map get_current_operations (session ref
    session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> network_operations)map`

value of the field

RPC name: get_default_locking_mode *Overview:*

Get the default_locking_mode field of the given network.

Signature:

```
network_default_locking_mode get_default_locking_mode (session ref
    session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network_default_locking_mode`

value of the field

RPC name: get_managed *Overview:*

Get the managed field of the given network.

Signature:

```
bool get_managed (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_MTU *Overview:*

Get the MTU field of the given network.

Signature:

```
int get_MTU (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given network.

Signature:

```
string get_name_description (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given network.

Signature:

```
string get_name_label (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given network.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PIFs *Overview:*

Get the PIFs field of the given network.

Signature:

```
PIF ref set get_PIFs (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref set`

value of the field

RPC name: get_purpose *Overview:*

Get the purpose field of the given network.

Signature:

```
network_purpose set get_purpose (session ref session_id, network ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network_purpose set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given network.

Signature:

```
network record get_record (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network record`

all fields from the object

RPC name: get_tags *Overview:*

Get the tags field of the given network.

Signature:

```
string set get_tags (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given network.

Signature:

```
string get_uuid (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VIFs *Overview:*

Get the VIFs field of the given network.

Signature:

```
VIF ref set get_VIFs (session ref session_id, network ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VIF ref set`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given network. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, network ref self
    , string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_purpose *Overview:*

Remove a purpose from a network (if present)

Signature:

```
void remove_purpose (session ref session_id, network ref self,
    network_purpose value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	The network
`network_purpose`	value	The purpose to remove

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: remove_tags *Overview:*

Remove the given value from the tags field of the given network. If the value is not in that Set, then do nothing.

Signature:

```
void remove_tags (session ref session_id, network ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`string`	value	Value to remove

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: set_default_locking_mode *Overview:*

Set the default locking mode for VIFs attached to this network

Signature:

```
void set_default_locking_mode (session ref session_id, network ref
    network, network_default_locking_mode value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	network	The network
`network_default_locking_mode`	value	The default locking mode for VIFs attached to this network.

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_MTU *Overview:*

Set the MTU field of the given network.

Signature:

```
void set_MTU (session ref session_id, network ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`int`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_name_description *Overview:*

Set the name/description field of the given network.

Signature:

```
void set_name_description (session ref session_id, network ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name/label field of the given network.

Signature:

```
void set_name_label (session ref session_id, network ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given network.

Signature:

```
void set_other_config (session ref session_id, network ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
network ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_tags *Overview:*

Set the tags field of the given network.

Signature:

```
void set_tags (session ref session_id, network ref self, string set
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
network ref	self	reference to the object
string set	value	New value to set

Minimum Role: vm-operator

Return Type: **void**

Class: network_sriov

network-sriov which connects logical pif and physical pif

Fields for class: network_sriov

Field	Type	Qualifier	Description
configuration_mode	`sriov_configuration_mode`	*RO/runtime*	The mode for configure network sriov
logical_PIF	`PIF ref`	*RO/constructor*	The logical PIF to connect to the SR-IOV network after enable SR-IOV on the physical PIF
physical_PIF	`PIF ref`	*RO/constructor*	The PIF that has SR-IOV enabled
requires_reboot	`bool`	*RO/runtime*	Indicates whether the host need to be rebooted before SR-IOV is enabled on the physical PIF
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: network_sriov

RPC name: create *Overview:*

Enable SR-IOV on the specific PIF. It will create a network-sriov based on the specific PIF and automatically create a logical PIF to connect the specific network.

Signature:

```
network_sriov ref create (session ref session_id, PIF ref pif, network
    ref network)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	pif	PIF on which to enable SR-IOV
`network ref`	network	Network to connect SR-IOV virtual functions with VM VIFs

Minimum Role: pool-operator

Return Type: `network_sriov ref`

The reference of the created network_sriov object

RPC name: destroy *Overview:*

Disable SR-IOV on the specific PIF. It will destroy the network-sriov and the logical PIF accordingly.

Signature:

```
void destroy (session ref session_id, network_sriov ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	SRIOV to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the network_sriovs known to the system.

Signature:

```
network_sriov ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `network_sriov ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of network_sriov references to network_sriov records for all network_sriovs known to the system.

Signature:

```
(network_sriov ref -> network_sriov record) map get_all_records (
    session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(network_sriov ref -> network_sriov record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the network_sriov instance with the specified UUID.

Signature:

```
network_sriov ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `network_sriov ref`

reference to the object

RPC name: get_configuration_mode *Overview:*

Get the configuration_mode field of the given network_sriov.

Signature:

```
sriov_configuration_mode get_configuration_mode (session ref session_id
    , network_sriov ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	reference to the object

Minimum Role: read-only

Return Type: `sriov_configuration_mode`

value of the field

RPC name: get_logical_PIF *Overview:*

Get the logical_PIF field of the given network_sriov.

Signature:

```
PIF ref get_logical_PIF (session ref session_id, network_sriov ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_physical_PIF *Overview:*

Get the physical_PIF field of the given network_sriov.

Signature:

```
PIF ref get_physical_PIF (session ref session_id, network_sriov ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given network_sriov.

Signature:

```
network_sriov record get_record (session ref session_id, network_sriov
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network_sriov record`

all fields from the object

RPC name: get_remaining_capacity *Overview:*

Get the number of free SR-IOV VFs on the associated PIF

Signature:

```
int get_remaining_capacity (session ref session_id, network_sriov ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	the NETWORK_SRIOV object

Minimum Role: read-only

Return Type: **`int`**

The number of free SR-IOV VFs on the associated PIF

RPC name: get_requires_reboot *Overview:*

Get the requires_reboot field of the given network_sriov.

Signature:

```
bool get_requires_reboot (session ref session_id, network_sriov ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given network_sriov.

Signature:

```
string get_uuid (session ref session_id, network_sriov ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network_sriov ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: Observer

Describes a observer which will control observability activity in the Toolstack

Fields for class: Observer

Field	Type	Qualifier	Description
attributes	`(string -> string)map`	*RO/constructor*	Attributes that observer will add to the data they produce
components	`string set`	*RO/constructor*	The list of xenserver components the observer will broadcast. An empty list means all components
enabled	`bool`	*RO/constructor*	This denotes if the observer is enabled. true if it is enabled and false if it is disabled

Field	Type	Qualifier	Description
endpoints	`string set`	*RO/constructor*	The list of endpoints where data is exported to. Each endpoint is a URL or the string 'bugtool' refering to the internal logs
hosts	`host ref set`	*RO/constructor*	The list of hosts the observer is active on. An empty list means all hosts
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: Observer

RPC name: create *Overview:*

Create a new Observer instance, and return its handle.

Signature:

```
Observer ref create (session ref session_id, Observer record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer record`	args	All constructor arguments

Minimum Role: pool-admin

Return Type: `Observer ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified Observer instance.

Signature:

```
void destroy (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: pool-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the Observers known to the system.

Signature:

```
Observer ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Observer ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Observer references to Observer records for all Observers known to the system.

Signature:

```
(Observer ref -> Observer record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Observer ref -> Observer record)map`

records of all objects

RPC name: get_attributes *Overview:*

Get the attributes field of the given Observer.

Signature:

```
(string -> string) map get_attributes (session ref session_id, Observer
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the Observer instances with the given label.

Signature:

```
Observer ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `Observer ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the Observer instance with the specified UUID.

Signature:

```
Observer ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Observer ref`

reference to the object

RPC name: get_components *Overview:*

Get the components field of the given Observer.

Signature:

```
string set get_components (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_enabled *Overview:*

Get the enabled field of the given Observer.

Signature:

```
bool get_enabled (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_endpoints *Overview:*

Get the endpoints field of the given Observer.

Signature:

```
string set get_endpoints (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_hosts *Overview:*

Get the hosts field of the given Observer.

Signature:

```
host ref set get_hosts (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref set`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given Observer.

Signature:

```
string get_name_description (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given Observer.

Signature:

```
string get_name_label (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Observer.

Signature:

```
Observer record get_record (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Observer record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given Observer.

Signature:

```
string get_uuid (session ref session_id, Observer ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: set_attributes *Overview:*

Set the attributes of an observer. These are used to emit metadata by the observer

Signature:

```
void set_attributes (session ref session_id, Observer ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	The observer
`(string -> string)map`	value	The attributes that the observer emits as part of the data

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_components *Overview:*

Set the components on which the observer will broadcast to. i.e. xapi, xenopsd, networkd, etc

Signature:

```
void set_components (session ref session_id, Observer ref self, string
    set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	The observer
`string set`	value	The components the observer will broadcast to

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_enabled *Overview:*

Enable / disable this observer which will stop the observer from producing observability information

Signature:

```
void set_enabled (session ref session_id, Observer ref self, bool value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	The observer
`bool`	value	If the observer is to be enabled (true) or disabled (false)

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_endpoints *Overview:*

Set the file/HTTP endpoints the observer sends data to

Signature:

```
void set_endpoints (session ref session_id, Observer ref self, string
    set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	The observer
`string set`	value	The endpoints that the observer will export data to. A URL or the string 'bugtool'. This can refer to an enpoint to the local file system

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_hosts *Overview:*

Sets the hosts that the observer is to be registered on

Signature:

```
void set_hosts (session ref session_id, Observer ref self, host ref set
     value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	The observer
`host ref set`	value	Hosts the observer is registered on

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_name_description *Overview:*

Set the name/description field of the given Observer.

Signature:

```
void set_name_description (session ref session_id, Observer ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_name_label *Overview:*

Set the name/label field of the given Observer.

Signature:

```
void set_name_label (session ref session_id, Observer ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Observer ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

Class: PBD

The physical block devices through which hosts access SRs

Fields for class: PBD

Field	Type	Qualifier	Description
currently_attached	`bool`	*RO/runtime*	is the SR currently attached on this host?
device_config	`(string -> string)map`	*RO/constructor*	a config string to string map that is provided to the host's SR-backend-driver
host	`host ref`	*RO/constructor*	physical machine on which the pbd is available
other_config	`(string -> string)map`	*RW*	additional configuration
SR	`SR ref`	*RO/constructor*	the storage repository that the pbd realises
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: PBD

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given PBD.

Signature:

```
void add_to_other_config (session ref session_id, PBD ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PBD ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: pool-operator

Return Type: **void**

RPC name: create *Overview:*

Create a new PBD instance, and return its handle.

Signature:

```
PBD ref create (session ref session_id, PBD record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PBD record	args	All constructor arguments

Minimum Role: pool-operator

Return Type: PBD ref

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified PBD instance.

Signature:

```
void destroy (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the PBDs known to the system.

Signature:

```
PBD ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PBD ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PBD references to PBD records for all PBDs known to the system.

Signature:

```
(PBD ref -> PBD record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PBD ref -> PBD record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PBD instance with the specified UUID.

Signature:

```
PBD ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PBD ref`

reference to the object

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given PBD.

Signature:

```
bool get_currently_attached (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_device_config *Overview:*

Get the device_config field of the given PBD.

Signature:

```
(string -> string) map get_device_config (session ref session_id, PBD
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given PBD.

Signature:

```
host ref get_host (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given PBD.

Signature:

```
(string -> string) map get_other_config (session ref session_id, PBD
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PBD.

Signature:

```
PBD record get_record (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PBD record`

all fields from the object

RPC name: get_SR *Overview:*

Get the SR field of the given PBD.

Signature:

```
SR ref get_SR (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PBD.

Signature:

```
string get_uuid (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: plug *Overview:*

Activate the specified PBD, causing the referenced SR to be attached and scanned

Signature:

```
void plug (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	The PBD to activate

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `SR_UNKNOWN_DRIVER`

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given PBD. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, PBD ref self,
   string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_device_config *Overview:*

Sets the PBD's device_config field

Signature:

```
void set_device_config (session ref session_id, PBD ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	The PBD to modify
`(string -> string)map`	value	The new value of the PBD's device_config

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given PBD.

Signature:

```
void set_other_config (session ref session_id, PBD ref self, (string ->
     string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: unplug *Overview:*

Deactivate the specified PBD, causing the referenced SR to be detached and nolonger scanned

Signature:

```
void unplug (session ref session_id, PBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PBD ref`	self	The PBD to deactivate

Minimum Role: pool-operator

Return Type: `void`

Class: PCI

A PCI device

Fields for class: PCI

Field	Type	Qualifier	Description
class_name	`string`	*RO/constructor*	PCI class name
dependencies	`PCI ref set`	*RO/runtime*	List of dependent PCI devices
device_name	`string`	*RO/constructor*	Device name
driver_name	`string`	*RO/constructor*	Driver name
host	`host ref`	*RO/constructor*	Physical machine that owns the PCI device
other_config	`(string -> string)map`	*RW*	Additional configuration
pci_id	`string`	*RO/constructor*	PCI ID of the physical device
subsystem_device_name	`string`	*RO/constructor*	Subsystem device name
subsystem_vendor_name	`string`	*RO/constructor*	Subsystem vendor name

Field	Type	Qualifier	Description
uuid	`string`	*RO/runtime*	Unique identifier/object reference
vendor_name	`string`	*RO/constructor*	Vendor name

RPCs associated with class: PCI

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given PCI.

Signature:

```
void add_to_other_config (session ref session_id, PCI ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the PCIs known to the system.

Signature:

```
PCI ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PCI ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PCI references to PCI records for all PCIs known to the system.

Signature:

```
(PCI ref -> PCI record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PCI ref -> PCI record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PCI instance with the specified UUID.

Signature:

```
PCI ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PCI ref`

reference to the object

RPC name: get_class_name *Overview:*

Get the class_name field of the given PCI.

Signature:

```
string get_class_name (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_dependencies *Overview:*

Get the dependencies field of the given PCI.

Signature:

```
PCI ref set get_dependencies (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI ref set`

value of the field

RPC name: get_device_name *Overview:*

Get the device_name field of the given PCI.

Signature:

```
string get_device_name (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_driver_name *Overview:*

Get the driver_name field of the given PCI.

Signature:

```
string get_driver_name (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given PCI.

Signature:

```
host ref get_host (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given PCI.

Signature:

```
(string -> string) map get_other_config (session ref session_id, PCI
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_pci_id *Overview:*

Get the pci_id field of the given PCI.

Signature:

```
string get_pci_id (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PCI.

Signature:

```
PCI record get_record (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI record`

all fields from the object

RPC name: get_subsystem_device_name *Overview:*

Get the subsystem_device_name field of the given PCI.

Signature:

```
string get_subsystem_device_name (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_subsystem_vendor_name *Overview:*

Get the subsystem_vendor_name field of the given PCI.

Signature:

```
string get_subsystem_vendor_name (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PCI.

Signature:

```
string get_uuid (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor_name *Overview:*

Get the vendor_name field of the given PCI.

Signature:

```
string get_vendor_name (session ref session_id, PCI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given PCI. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, PCI ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given PCI.

Signature:

```
void set_other_config (session ref session_id, PCI ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PCI ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: PGPU

A physical GPU (pGPU)

Fields for class: PGPU

Field	Type	Qualifier	Description
compatibility_metadata	(string -> string) map	*RO/runtime*	PGPU metadata to determine whether a VGPU can migrate between two PGPUs
dom0_access	pgpu_dom0_access	*RO/runtime*	The accessibility of this device from dom0
enabled_VGPU_types	VGPU_type ref set	*RO/runtime*	List of VGPU types which have been enabled for this PGPU
GPU_group	GPU_group ref	*RO/constructor*	GPU group the pGPU is contained in
host	host ref	*RO/runtime*	Host that owns the GPU
is_system_display_device	bool	*RO/runtime*	Is this device the system display device
other_config	(string -> string) map	*RW*	Additional configuration
PCI	PCI ref	*RO/constructor*	Link to underlying PCI device
resident_VGPUs	VGPU ref set	*RO/runtime*	List of VGPUs running on this PGPU
supported_VGPU_max_capacities	(VGPU_type ref -> int) map	*RO/runtime*	A map relating each VGPU type supported on this GPU to the maximum number of VGPUs of that type which can run simultaneously on this GPU
supported_VGPU_types	VGPU_type ref set	*RO/runtime*	List of VGPU types supported by the underlying hardware
uuid	string	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: PGPU

RPC name: add_enabled_VGPU_types *Overview:*

Signature:

```
void add_enabled_VGPU_types (session ref session_id, PGPU ref self,
    VGPU_type ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PGPU ref	self	The PGPU to which we are adding an enabled VGPU type
VGPU_type ref	value	The VGPU type to enable

Minimum Role: pool-operator

Return Type: **void**

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given PGPU.

Signature:

```
void add_to_other_config (session ref session_id, PGPU ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PGPU ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: pool-operator

Return Type: **void**

RPC name: disable_dom0_access *Overview:*

Signature:

```
pgpu_dom0_access disable_dom0_access (session ref session_id, PGPU ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	The PGPU to which dom0 will be denied access

Minimum Role: pool-operator

Return Type: `pgpu_dom0_access`

The accessibility of this PGPU from dom0

RPC name: enable_dom0_access *Overview:*

Signature:

```
pgpu_dom0_access enable_dom0_access (session ref session_id, PGPU ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	The PGPU to which dom0 will be granted access

Minimum Role: pool-operator

Return Type: `pgpu_dom0_access`

The accessibility of this PGPU from dom0

RPC name: get_all *Overview:*

Return a list of all the PGPUs known to the system.

Signature:

```
PGPU ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PGPU ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PGPU references to PGPU records for all PGPUs known to the system.

Signature:

```
(PGPU ref -> PGPU record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PGPU ref -> PGPU record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PGPU instance with the specified UUID.

Signature:

```
PGPU ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PGPU ref`

reference to the object

RPC name: get_compatibility_metadata *Overview:*

Get the compatibility_metadata field of the given PGPU.

Signature:

```
(string -> string) map get_compatibility_metadata (session ref
    session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_dom0_access *Overview:*

Get the dom0_access field of the given PGPU.

Signature:

```
pgpu_dom0_access get_dom0_access (session ref session_id, PGPU ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pgpu_dom0_access`

value of the field

RPC name: get_enabled_VGPU_types *Overview:*

Get the enabled_VGPU_types field of the given PGPU.

Signature:

```
VGPU_type ref set get_enabled_VGPU_types (session ref session_id, PGPU
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type ref set`

value of the field

RPC name: get_GPU_group *Overview:*

Get the GPU_group field of the given PGPU.

Signature:

```
GPU_group ref get_GPU_group (session ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `GPU_group ref`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given PGPU.

Signature:

```
host ref get_host (session ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_is_system_display_device *Overview:*

Get the is_system_display_device field of the given PGPU.

Signature:

```
bool get_is_system_display_device (session ref session_id, PGPU ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given PGPU.

Signature:

```
(string -> string) map get_other_config (session ref session_id, PGPU
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PCI *Overview:*

Get the PCI field of the given PGPU.

Signature:

```
PCI ref get_PCI (session ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI ref`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PGPU.

Signature:

```
PGPU record get_record (session ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU record`

all fields from the object

RPC name: get_remaining_capacity *Overview:*

Signature:

```
int get_remaining_capacity (session ref session_id, PGPU ref self,
    VGPU_type ref vgpu_type)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	The PGPU to query
`VGPU_type ref`	vgpu_type	The VGPU type for which we want to find the number of VGPUs which can still be started on this PGPU

Minimum Role: read-only

Return Type: `int`

The number of VGPUs of the specified type which can still be started on this PGPU

RPC name: get_resident_VGPUs *Overview:*

Get the resident_VGPUs field of the given PGPU.

Signature:

```
VGPU ref set get_resident_VGPUs (session ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU ref set`

value of the field

RPC name: get_supported_VGPU_max_capacities *Overview:*

Get the supported_VGPU_max_capacities field of the given PGPU.

Signature:

```
(VGPU_type ref -> int) map get_supported_VGPU_max_capacities (session
    ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(VGPU_type ref -> int)map`

value of the field

RPC name: get_supported_VGPU_types *Overview:*

Get the supported_VGPU_types field of the given PGPU.

Signature:

```
VGPU_type ref set get_supported_VGPU_types (session ref session_id,
    PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PGPU.

Signature:

```
string get_uuid (session ref session_id, PGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_enabled_VGPU_types *Overview:*

Signature:

```
void remove_enabled_VGPU_types (session ref session_id, PGPU ref self,
    VGPU_type ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	The PGPU from which we are removing an enabled VGPU type
`VGPU_type ref`	value	The VGPU type to disable

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given PGPU. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, PGPU ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_enabled_VGPU_types *Overview:*

Signature:

```
void set_enabled_VGPU_types (session ref session_id, PGPU ref self,
    VGPU_type ref set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	The PGPU on which we are enabling a set of VGPU types
`VGPU_type ref set`	value	The VGPU types to enable

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_GPU_group *Overview:*

Signature:

```
void set_GPU_group (session ref session_id, PGPU ref self, GPU_group
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	The PGPU to move to a new group
`GPU_group ref`	value	The group to which the PGPU will be moved

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given PGPU.

Signature:

```
void set_other_config (session ref session_id, PGPU ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PGPU ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: PIF

A physical network interface (note separate VLANs are represented as several PIFs)

Fields for class: PIF

Field	Type	Qualifier	Description
bond_master_of	`Bond ref set`	*RO/runtime*	Indicates this PIF represents the results of a bond
bond_slave_of	`Bond ref`	*RO/runtime*	Indicates which bond this interface is part of
capabilities	`string set`	*RO/runtime*	Additional capabilities on the interface.
currently_attached	`bool`	*RO/runtime*	true if this interface is online

Field	Type	Qualifier	Description
device	`string`	*RO/constructor*	machine-readable name of the interface (e.g. eth0)
disallow_unplug	`bool`	*RO/runtime*	Prevent this PIF from being unplugged; set this to notify the management tool-stack that the PIF has a special use and should not be unplugged under any circumstances (e.g. because you're running storage traffic over it)
DNS	`string`	*RO/runtime*	Comma separated list of the IP addresses of the DNS servers to use
gateway	`string`	*RO/runtime*	IP gateway
host	`host ref`	*RO/constructor*	physical machine to which this pif is connected
igmp_snooping_status	`pif_igmp_status`	*RO/runtime*	The IGMP snooping status of the corresponding network bridge
IP	`string`	*RO/runtime*	IP address
ip_configuration_mode	`ip_configuration_mode`	*RO/runtime*	Sets if and how this interface gets an IP address
IPv6	`string set`	*RO/runtime*	IPv6 address
ipv6_configuration_mode	`ipv6_configuration_mode`	*RO/runtime*	Sets if and how this interface gets an IPv6 address
ipv6_gateway	`string`	*RO/runtime*	IPv6 gateway

Field	Type	Qualifier	Description
MAC	`string`	*RO/constructor*	ethernet MAC address of physical interface
managed	`bool`	*RO/constructor*	Indicates whether the interface is managed by xapi. If it is not, then xapi will not configure the interface, the commands PIF.plug/unplug/reconfigure_ip(v6) cannot be used, nor can the interface be bonded or have VLANs based on top through xapi.
management	`bool`	*RO/runtime*	Indicates whether the control software is listening for connections on this interface
metrics	`PIF_metrics ref`	*RO/runtime*	metrics associated with this PIF
MTU	`int`	*RO/constructor*	MTU in octets
netmask	`string`	*RO/runtime*	IP netmask
network	`network ref`	*RO/constructor*	virtual network to which this pif is connected
other_config	`(string -> string)map`	*RW*	Additional configuration
PCI	`PCI ref`	*RO/runtime*	Link to underlying PCI device
physical	`bool`	*RO/runtime*	true if this represents a physical network interface
primary_address_type	`primary_address_type`	*RO/runtime*	Which protocol should define the primary address of this interface

Field	Type	Qualifier	Description
properties	`(string -> string)map`	*RO/runtime*	Additional configuration properties for the interface.
sriov_logical_PIF_of	`network_sriov ref set`	*RO/runtime*	Indicates which network_sriov this interface is logical of
sriov_physical_PIF_of	`network_sriov ref set`	*RO/runtime*	Indicates which network_sriov this interface is physical of
tunnel_access_PIF_of	`tunnel ref set`	*RO/runtime*	Indicates to which tunnel this PIF gives access
tunnel_transport_PIF_of	`tunnel ref set`	*RO/runtime*	Indicates to which tunnel this PIF provides transport
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VLAN	`int`	*RO/constructor*	VLAN tag for all traffic passing through this interface
VLAN_master_of	`VLAN ref`	*RO/runtime*	Indicates which VLAN this interface receives untagged traffic from
VLAN_slave_of	`VLAN ref set`	*RO/runtime*	Indicates which VLANs this interface transmits tagged traffic to

RPCs associated with class: PIF

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given PIF.

Signature:

```
void add_to_other_config (session ref session_id, PIF ref self, string
    key, string value)
```

```
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: `void`

RPC name: create_VLAN **This message is deprecated.**

Overview:

Create a VLAN interface from an existing physical interface. This call is deprecated: use VLAN.create instead

Signature:

```
PIF ref create_VLAN (session ref session_id, string device, network ref
    network, host ref host, int VLAN)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	device	physical interface on which to create the VLAN interface
`network ref`	network	network to which this interface should be connected
`host ref`	host	physical machine to which this PIF is connected
`int`	VLAN	VLAN tag for the new interface

Minimum Role: pool-operator

Return Type: `PIF ref`

The reference of the created PIF object

Possible Error Codes: `VLAN_TAG_INVALID`

RPC name: db_forget *Overview:*

Destroy a PIF database record.

Signature:

```
void db_forget (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	The ref of the PIF whose database record should be destroyed

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: db_introduce *Overview:*

Create a new PIF record in the database only

Signature:

```
PIF ref db_introduce (session ref session_id, string device, network
    ref network, host ref host, string MAC, int MTU, int VLAN, bool
    physical, ip_configuration_mode ip_configuration_mode, string IP,
    string netmask, string gateway, string DNS, Bond ref bond_slave_of,
    VLAN ref VLAN_master_of, bool management, (string -> string) map
    other_config, bool disallow_unplug, ipv6_configuration_mode
    ipv6_configuration_mode, string set IPv6, string ipv6_gateway,
    primary_address_type primary_address_type, bool managed, (string ->
    string) map properties)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	device	
`network ref`	network	
`host ref`	host	
`string`	MAC	
`int`	MTU	
`int`	VLAN	
`bool`	physical	
`ip_configuration_mode`	ip_configuration_mode	
`string`	IP	
`string`	netmask	
`string`	gateway	
`string`	DNS	
`Bond ref`	bond_slave_of	
`VLAN ref`	VLAN_master_of	
`bool`	management	
`(string -> string)map`	other_config	
`bool`	disallow_unplug	
`ipv6_configuration_mode`	ipv6_configuration_mode	
`string set`	IPv6	
`string`	ipv6_gateway	
`primary_address_type`	primary_address_type	
`bool`	managed	
`(string -> string)map`	properties	

Minimum Role: pool-operator

Return Type: `PIF ref`

The ref of the newly created PIF record.

RPC name: destroy **This message is deprecated.**

Overview:

Destroy the PIF object (provided it is a VLAN interface). This call is deprecated: use VLAN.destroy or Bond.destroy instead

Signature:

```
void destroy (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PIF ref	self	the PIF object to destroy

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `PIF_IS_PHYSICAL`

RPC name: forget *Overview:*

Destroy the PIF object matching a particular network interface

Signature:

```
void forget (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PIF ref	self	The PIF object to destroy

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `PIF_TUNNEL_STILL_EXISTS`, `CLUSTERING_ENABLED`

RPC name: get_all *Overview:*

Return a list of all the PIFs known to the system.

Signature:

```
PIF ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PIF ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PIF references to PIF records for all PIFs known to the system.

Signature:

```
(PIF ref -> PIF record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PIF ref -> PIF record)map`

records of all objects

RPC name: get_bond_master_of *Overview:*

Get the bond_master_of field of the given PIF.

Signature:

```
Bond ref set get_bond_master_of (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Bond ref set`

value of the field

RPC name: get_bond_slave_of *Overview:*

Get the bond_slave_of field of the given PIF.

Signature:

```
Bond ref get_bond_slave_of (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Bond ref`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the PIF instance with the specified UUID.

Signature:

```
PIF ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PIF ref`

reference to the object

RPC name: get_capabilities *Overview:*

Get the capabilities field of the given PIF.

Signature:

```
string set get_capabilities (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given PIF.

Signature:

```
bool get_currently_attached (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_device *Overview:*

Get the device field of the given PIF.

Signature:

```
string get_device (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_disallow_unplug *Overview:*

Get the disallow_unplug field of the given PIF.

Signature:

```
bool get_disallow_unplug (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_DNS *Overview:*

Get the DNS field of the given PIF.

Signature:

```
string get_DNS (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_gateway *Overview:*

Get the gateway field of the given PIF.

Signature:

```
string get_gateway (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given PIF.

Signature:

```
host ref get_host (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_igmp_snooping_status *Overview:*

Get the igmp_snooping_status field of the given PIF.

Signature:

```
pif_igmp_status get_igmp_snooping_status (session ref session_id, PIF
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pif_igmp_status`

value of the field

RPC name: get_IP *Overview:*

Get the IP field of the given PIF.

Signature:

```
string get_IP (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_ip_configuration_mode *Overview:*

Get the ip_configuration_mode field of the given PIF.

Signature:

```
ip_configuration_mode get_ip_configuration_mode (session ref session_id
    , PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `ip_configuration_mode`

value of the field

RPC name: get_IPv6 *Overview:*

Get the IPv6 field of the given PIF.

Signature:

```
string set get_IPv6 (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_ipv6_configuration_mode *Overview:*

Get the ipv6_configuration_mode field of the given PIF.

Signature:

```
ipv6_configuration_mode get_ipv6_configuration_mode (session ref
    session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `ipv6_configuration_mode`

value of the field

RPC name: get_ipv6_gateway *Overview:*

Get the ipv6_gateway field of the given PIF.

Signature:

```
string get_ipv6_gateway (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_MAC *Overview:*

Get the MAC field of the given PIF.

Signature:

```
string get_MAC (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_managed *Overview:*

Get the managed field of the given PIF.

Signature:

```
bool get_managed (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_management *Overview:*

Get the management field of the given PIF.

Signature:

```
bool get_management (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_metrics *Overview:*

Get the metrics field of the given PIF.

Signature:

```
PIF_metrics ref get_metrics (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF_metrics ref`

value of the field

RPC name: get_MTU *Overview:*

Get the MTU field of the given PIF.

Signature:

```
int get_MTU (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_netmask *Overview:*

Get the netmask field of the given PIF.

Signature:

```
string get_netmask (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_network *Overview:*

Get the network field of the given PIF.

Signature:

```
network ref get_network (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given PIF.

Signature:

```
(string -> string) map get_other_config (session ref session_id, PIF
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PCI *Overview:*

Get the PCI field of the given PIF.

Signature:

```
PCI ref get_PCI (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI ref`

value of the field

RPC name: get_physical *Overview:*

Get the physical field of the given PIF.

Signature:

```
bool get_physical (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_primary_address_type *Overview:*

Get the primary_address_type field of the given PIF.

Signature:

```
primary_address_type get_primary_address_type (session ref session_id,
    PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `primary_address_type`

value of the field

RPC name: get_properties *Overview:*

Get the properties field of the given PIF.

Signature:

```
(string -> string) map get_properties (session ref session_id, PIF ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PIF.

Signature:

```
PIF record get_record (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF record`

all fields from the object

RPC name: get_sriov_logical_PIF_of *Overview:*

Get the sriov_logical_PIF_of field of the given PIF.

Signature:

```
network_sriov ref set get_sriov_logical_PIF_of (session ref session_id,
    PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network_sriov ref set`

value of the field

RPC name: get_sriov_physical_PIF_of *Overview:*

Get the sriov_physical_PIF_of field of the given PIF.

Signature:

```
network_sriov ref set get_sriov_physical_PIF_of (session ref session_id
   , PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network_sriov ref set`

value of the field

RPC name: get_tunnel_access_PIF_of *Overview:*

Get the tunnel_access_PIF_of field of the given PIF.

Signature:

```
tunnel ref set get_tunnel_access_PIF_of (session ref session_id, PIF
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `tunnel ref set`

value of the field

RPC name: get_tunnel_transport_PIF_of *Overview:*

Get the tunnel_transport_PIF_of field of the given PIF.

Signature:

```
tunnel ref set get_tunnel_transport_PIF_of (session ref session_id, PIF
     ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `tunnel ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PIF.

Signature:

```
string get_uuid (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VLAN *Overview:*

Get the VLAN field of the given PIF.

Signature:

```
int get_VLAN (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_VLAN_master_of *Overview:*

Get the VLAN_master_of field of the given PIF.

Signature:

```
VLAN ref get_VLAN_master_of (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VLAN ref`

value of the field

RPC name: get_VLAN_slave_of *Overview:*

Get the VLAN_slave_of field of the given PIF.

Signature:

```
VLAN ref set get_VLAN_slave_of (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VLAN ref set`

value of the field

RPC name: introduce *Overview:*

Create a PIF object matching a particular network interface

Signature:

```
PIF ref introduce (session ref session_id, host ref host, string MAC,
    string device, bool managed)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host on which the interface exists
`string`	MAC	The MAC address of the interface
`string`	device	The device name to use for the interface
`bool`	managed	Indicates whether the interface is managed by xapi (defaults to "true")

Minimum Role: pool-operator

Return Type: `PIF ref`

The reference of the created PIF object

RPC name: plug *Overview:*

Attempt to bring up a physical interface

Signature:

```
void plug (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
PIF ref	self	the PIF object to plug

Minimum Role: pool-operator

Return Type: **void**

Possible Error Codes: TRANSPORT_PIF_NOT_CONFIGURED

RPC name: reconfigure_ip *Overview:*

Reconfigure the IP address settings for this interface

Signature:

```
void reconfigure_ip (session ref session_id, PIF ref self,
    ip_configuration_mode mode, string IP, string netmask, string
    gateway, string DNS)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PIF ref	self	the PIF object to reconfigure
ip_configuration_mode	mode	whether to use dynamic/static/no-assignment
string	IP	the new IP address
string	netmask	the new netmask
string	gateway	the new gateway
string	DNS	the new DNS settings

Minimum Role: pool-operator

Return Type: **void**

Possible Error Codes: CLUSTERING_ENABLED

RPC name: reconfigure_ipv6 *Overview:*

Reconfigure the IPv6 address settings for this interface

Signature:

```
void reconfigure_ipv6 (session ref session_id, PIF ref self,
    ipv6_configuration_mode mode, string IPv6, string gateway, string
    DNS)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	the PIF object to reconfigure
`ipv6_configuration_mode`	mode	whether to use dynamic/static/no-assignment
`string`	IPv6	the new IPv6 address (in / format)
`string`	gateway	the new gateway
`string`	DNS	the new DNS settings

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `CLUSTERING_ENABLED`

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given PIF. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, PIF ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PIF ref	self	reference to the object
string	key	Key to remove

Minimum Role: pool-operator

Return Type: **void**

RPC name: scan *Overview:*

Scan for physical interfaces on a host and create PIF objects to represent them

Signature:

```
void scan (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host on which to scan

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_disallow_unplug *Overview:*

Set whether unplugging the PIF is allowed

Signature:

```
void set_disallow_unplug (session ref session_id, PIF ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	Reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `OTHER_OPERATION_IN_PROGRESS`, `CLUSTERING_ENABLED`

RPC name: set_other_config *Overview:*

Set the other_config field of the given PIF.

Signature:

```
void set_other_config (session ref session_id, PIF ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_primary_address_type *Overview:*

Change the primary address type used by this PIF

Signature:

```
void set_primary_address_type (session ref session_id, PIF ref self,
    primary_address_type primary_address_type)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PIF ref	self	the PIF object to reconfigure
primary_address_type	primary_address_type	Whether to prefer IPv4 or IPv6 connections

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_property *Overview:*

Set the value of a property of the PIF

Signature:

```
void set_property (session ref session_id, PIF ref self, string name,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PIF ref	self	The PIF
string	name	The property name
string	value	The property value

Minimum Role: pool-operator

Return Type: **void**

RPC name: unplug *Overview:*

Attempt to bring down a physical interface

Signature:

```
void unplug (session ref session_id, PIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	self	the PIF object to unplug

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `HA_OPERATION_WOULD_BREAK_FAILOVER_PLAN`, `VIF_IN_USE`, `PIF_DOES_NOT_ALLOW_UNPLUG`, `PIF_HAS_FCOE_SR_IN_USE`

Class: PIF_metrics

The metrics associated with a physical network interface

Fields for class: PIF_metrics

Field	Type	Qualifier	Description
carrier	`bool`	*RO/runtime*	Report if the PIF got a carrier or not
device_id	`string`	*RO/runtime*	Report device ID
device_name	`string`	*RO/runtime*	Report device name
duplex	`bool`	*RO/runtime*	Full duplex capability of the link (if available)
io_read_kbs	**`float`**	*RO/runtime*	**Removed**. Read bandwidth (KiB/s)
io_write_kbs	**`float`**	*RO/runtime*	**Removed**. Write bandwidth (KiB/s)
last_updated	`datetime`	*RO/runtime*	Time at which this information was last updated

Field	Type	Qualifier	Description
other_config	`(string -> string) map`	*RW*	additional configuration
pci_bus_path	`string`	*RO/runtime*	PCI bus path of the pif (if available)
speed	`int`	*RO/runtime*	Speed of the link (if available)
uuid	`string`	*RO/runtime*	Unique identifier/object reference
vendor_id	`string`	*RO/runtime*	Report vendor ID
vendor_name	`string`	*RO/runtime*	Report vendor name

RPCs associated with class: PIF_metrics

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given PIF_metrics.

Signature:

```
void add_to_other_config (session ref session_id, PIF_metrics ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the PIF_metrics instances known to the system.

Signature:

```
PIF_metrics ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PIF_metrics ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PIF_metrics references to PIF_metrics records for all PIF_metrics instances known to the system.

Signature:

```
(PIF_metrics ref -> PIF_metrics record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PIF_metrics ref -> PIF_metrics record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PIF_metrics instance with the specified UUID.

Signature:

```
PIF_metrics ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PIF_metrics ref`

reference to the object

RPC name: get_carrier *Overview:*

Get the carrier field of the given PIF_metrics.

Signature:

```
bool get_carrier (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_device_id *Overview:*

Get the device_id field of the given PIF_metrics.

Signature:

```
string get_device_id (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_device_name *Overview:*

Get the device_name field of the given PIF_metrics.

Signature:

```
string get_device_name (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_duplex *Overview:*

Get the duplex field of the given PIF_metrics.

Signature:

```
bool get_duplex (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_io_read_kbs This message is removed.

Overview:

Get the io/read_kbs field of the given PIF_metrics.

Signature:

```
float get_io_read_kbs (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `float`

value of the field

RPC name: get_io_write_kbs This message is removed.

Overview:

Get the io/write_kbs field of the given PIF_metrics.

Signature:

```
float get_io_write_kbs (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `float`

value of the field

RPC name: get_last_updated *Overview:*

Get the last_updated field of the given PIF_metrics.

Signature:

```
datetime get_last_updated (session ref session_id, PIF_metrics ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given PIF_metrics.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_pci_bus_path *Overview:*

Get the pci_bus_path field of the given PIF_metrics.

Signature:

```
string get_pci_bus_path (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PIF_metrics.

Signature:

```
PIF_metrics record get_record (session ref session_id, PIF_metrics ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF_metrics record`

all fields from the object

RPC name: get_speed *Overview:*

Get the speed field of the given PIF_metrics.

Signature:

```
int get_speed (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PIF_metrics.

Signature:

```
string get_uuid (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor_id *Overview:*

Get the vendor_id field of the given PIF_metrics.

Signature:

```
string get_vendor_id (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor_name *Overview:*

Get the vendor_name field of the given PIF_metrics.

Signature:

```
string get_vendor_name (session ref session_id, PIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given PIF_metrics. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, PIF_metrics ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given PIF_metrics.

Signature:

```
void set_other_config (session ref session_id, PIF_metrics ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF_metrics ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: pool

Pool-wide information

Fields for class: pool

Field	Type	Qualifier	Description
allowed_operations	`pool_allowed_operations set`	*RO/runtime*	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.
blobs	`(string -> blob ref)map`	*RO/runtime*	Binary blobs associated with this pool
client_certificate_auth_enabled	`bool`	*RO/runtime*	True if authentication by TLS client certificates is enabled
client_certificate_auth_name	`string`	*RO/runtime*	The name (CN/SAN) that an incoming client certificate must have to allow authentication
coordinator_bias	`bool`	*RW*	true if bias against pool master when scheduling vms is enabled, false otherwise
cpu_info	`(string -> string)map`	*RO/runtime*	Details about the physical CPUs on the pool
crash_dump_SR	`SR ref`	*RW*	The SR in which VDIs for crash dumps are created

Field	Type	Qualifier	Description
current_operations	`(string -> pool_allowed_operations )map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
custom_uefi_certificates	`string`	*RO/constructor*	Custom UEFI certificates allowing Secure Boot
default_SR	`SR ref`	*RW*	Default SR for VDIs
ext_auth_max_threads	**`int`**	*RO/constructor*	Maximum number of threads to use for external (AD) authentication
guest_agent_config	`(string -> string)map`	*RO/runtime*	Pool-wide guest agent configuration information
gui_config	`(string -> string)map`	*RW*	gui-specific configuration for pool
ha_allow_overcommit	`bool`	*RW*	If set to false then operations which would cause the Pool to become overcommitted will be blocked.
ha_cluster_stack	`string`	*RO/runtime*	The HA cluster stack that is currently in use. Only valid when HA is enabled.
ha_configuration	`(string -> string)map`	*RO/runtime*	The current HA configuration
ha_enabled	`bool`	*RO/runtime*	true if HA is enabled on the pool, false otherwise

Field	Type	Qualifier	Description
ha_host_failures_to_tolerate	int	*RO/runtime*	Number of host failures to tolerate before the Pool is declared to be overcommitted
ha_overcommitted	bool	*RO/runtime*	True if the Pool is considered to be overcommitted i.e. if there exist insufficient physical resources to tolerate the configured number of host failures
ha_plan_exists_for	int	*RO/runtime*	Number of future host failures we have managed to find a plan for. Once this reaches zero any future host failures will cause the failure of protected VMs.
ha_statefiles	string set	*RO/runtime*	HA statefile VDIs in use
health_check_config	(string -> string) map	*RW*	Configuration for the automatic health check feature
igmp_snooping_enabled	bool	*RO/runtime*	true if IGMP snooping is enabled in the pool, false otherwise.
is_psr_pending	bool	*RW*	True if either a PSR is running or we are waiting for a PSR to be re-run
last_update_sync	datetime	*RO/runtime*	time of the last update sychronization
live_patching_disabled	bool	*RW*	The pool-wide flag to show if the live patching feauture is disabled or not.

Field	Type	Qualifier	Description
local_auth_max_threads	`int`	*RO/constructor*	Maximum number of threads to use for PAM authentication
master	`host ref`	*RO/runtime*	The host that is pool master
metadata_VDIs	`VDI ref set`	*RO/runtime*	The set of currently known metadata VDIs for this pool
migration_compression	`bool`	*RW*	Default behaviour during migration, True if stream compression should be used
name_description	`string`	*RW*	Description
name_label	`string`	*RW*	Short name
other_config	`(string -> string)map`	*RW*	additional configuration
policy_no_vendor_device	`bool`	*RW*	The pool-wide policy for clients on whether to use the vendor device or not on newly created VMs. This field will also be consulted if the 'has_vendor_device' field is not specified in the VM.create call.
redo_log_enabled	`bool`	*RO/runtime*	true a redo-log is to be used other than when HA is enabled, false otherwise
redo_log_vdi	`VDI ref`	*RO/runtime*	indicates the VDI to use for the redo-log other than when HA is enabled
repositories	`Repository ref set`	*RO/runtime*	The set of currently enabled repositories

Field	Type	Qualifier	Description
repository_proxy_password	`secret ref`	*RO/runtime*	Password for the authentication of the proxy used in syncing with the enabled repositories
repository_proxy_url	`string`	*RO/runtime*	Url of the proxy used in syncing with the enabled repositories
repository_proxy_username	`string`	*RO/runtime*	Username for the authentication of the proxy used in syncing with the enabled repositories
restrictions	`(string -> string) map`	*RO/runtime*	Pool-wide restrictions currently in effect
suspend_image_SR	`SR ref`	*RW*	The SR in which VDIs for suspend images are created
tags	`string set`	*RW*	user-specified tags for categorization purposes
telemetry_frequency	`telemetry_frequency`	*RO/runtime*	How often the telemetry collection will be carried out
telemetry_next_collection	`datetime`	*RO/runtime*	The earliest timestamp (in UTC) when the next round of telemetry collection can be carried out
telemetry_uuid	`secret ref`	*RO/runtime*	The UUID of the pool for identification of telemetry data
tls_verification_enabled	`bool`	*RO/runtime*	True iff TLS certificate verification is enabled
uefi_certificates	`string`	*RO/constructor*	The UEFI certificates allowing Secure Boot

Field	Type	Qualifier	Description
update_sync_day	`int`	*RO/runtime*	The day of the week the update synchronizations will be scheduled, based on pool's local timezone. Ignored when update_sync_frequency is daily
update_sync_enabled	`bool`	*RO/runtime*	Whether periodic update synchronization is enabled or not
update_sync_frequency	`update_sync_frequency`	*RO/runtime*	The frequency at which updates are synchronized from a remote CDN: daily or weekly.
uuid	`string`	*RO/runtime*	Unique identifier/object reference
vswitch_controller	`string`	*RO/runtime*	**Deprecated**. address of the vswitch controller
wlb_enabled	`bool`	*RW*	true if workload balancing is enabled on the pool, false otherwise
wlb_url	`string`	*RO/runtime*	Url for the configured workload balancing host
wlb_username	`string`	*RO/runtime*	Username for accessing the workload balancing host

Field	Type	Qualifier	Description
wlb_verify_cert	`bool`	*RW*	**Deprecated**. true if communication with the WLB server should enforce TLS certificate verification.

RPCs associated with class: pool

RPC name: add_repository *Overview:*

Add a repository to the enabled set

Signature:

```
void add_repository (session ref session_id, pool ref self, Repository
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`Repository ref`	value	The repository to be added to the enabled set

Minimum Role: client-cert

Return Type: **`void`**

RPC name: add_tags *Overview:*

Add the given value to the tags field of the given pool. If the value is already in that Set, then do nothing.

Signature:

```
void add_tags (session ref session_id, pool ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool ref	self	reference to the object
string	value	New value to add

Minimum Role: vm-operator

Return Type: **void**

RPC name: add_to_guest_agent_config *Overview:*

Add a key-value pair to the pool-wide guest agent configuration

Signature:

```
void add_to_guest_agent_config (session ref session_id, pool ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool ref	self	The pool
string	key	The key to add
string	value	The value to add

Minimum Role: pool-admin

Return Type: **void**

RPC name: add_to_gui_config *Overview:*

Add the given key-value pair to the gui_config field of the given pool.

Signature:

```
void add_to_gui_config (session ref session_id, pool ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: add_to_health_check_config *Overview:*

Add the given key-value pair to the health_check_config field of the given pool.

Signature:

```
void add_to_health_check_config (session ref session_id, pool ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given pool.

Signature:

```
void add_to_other_config (session ref session_id, pool ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: apply_edition *Overview:*

Apply an edition to all hosts in the pool

Signature:

```
void apply_edition (session ref session_id, pool ref self, string
    edition)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	Reference to the pool
`string`	edition	The requested edition

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: certificate_install **This message is deprecated.**

Overview:

Install a TLS CA certificate, pool-wide.

Signature:

```
void certificate_install (session ref session_id, string name, string
    cert)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	A name to give the certificate
`string`	cert	The certificate in PEM format

Minimum Role: pool-operator

Return Type: `void`

RPC name: certificate_list **This message is deprecated.**

Overview:

List the names of all installed TLS CA certificates.

Signature:

```
string set certificate_list (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `string set`

All installed certificates

RPC name: certificate_sync *Overview:*

Copy the TLS CA certificates and CRLs of the master to all slaves.

Signature:

```
void certificate_sync (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: certificate_uninstall **This message is deprecated.**

Overview:

Remove a pool-wide TLS CA certificate.

Signature:

```
void certificate_uninstall (session ref session_id, string name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	The certificate name

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: check_update_readiness *Overview:*

Check if the pool is ready to be updated. If not, report the reasons.

Signature:

```
string set set check_update_readiness (session ref session_id, pool ref
    self, bool requires_reboot)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`bool`	requires_reboot	Assume that the update will require host reboots

Minimum Role: client-cert

Return Type: `string set set`

A set of error codes with arguments, if the pool is
not ready to update. An empty list means the pool can be updated.

RPC name: configure_repository_proxy *Overview:*

Configure proxy for RPM package repositories.

Signature:

```
void configure_repository_proxy (session ref session_id, pool ref self,
    string url, string username, string password)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`string`	url	The URL of the proxy server
`string`	username	The username used to authenticate with the proxy server
`string`	password	The password used to authenticate with the proxy server

Minimum Role: client-cert

Return Type: `void`

RPC name: configure_update_sync *Overview:*

Configure periodic update synchronization to sync updates from a remote CDN

Signature:

```
void configure_update_sync (session ref session_id, pool ref self,
    update_sync_frequency update_sync_frequency, int update_sync_day)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool

type	name	description
`update_sync_frequency`	update_sync_frequency	The frequency at which updates are synchronized from a remote CDN: daily or weekly.
`int`	update_sync_day	The day of the week the update synchronization will happen, based on pool's local timezone. Valid values are 0 to 6, 0 being Sunday. For 'daily' schedule, the value is ignored.

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create_new_blob *Overview:*

Create a placeholder for a named binary blob of data that is associated with this pool

Signature:

```
blob ref create_new_blob (session ref session_id, pool ref pool, string
    name, string mime_type, bool public)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	pool	The pool
`string`	name	The name associated with the blob
`string`	mime_type	The mime type for the data. Empty string translates to application/octet-stream
`bool`	public	True if the blob should be publicly available

Minimum Role: pool-operator

Return Type: `blob ref`

The reference of the blob, needed for populating its data

RPC name: create_VLAN *Overview:*

Create PIFs, mapping a network to the same physical interface/VLAN on each host. This call is deprecated: use Pool.create_VLAN_from_PIF instead.

Signature:

```
PIF ref set create_VLAN (session ref session_id, string device, network
    ref network, int VLAN)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	device	physical interface on which to create the VLAN interface
`network ref`	network	network to which this interface should be connected
`int`	VLAN	VLAN tag for the new interface

Minimum Role: pool-operator

Return Type: `PIF ref set`

The references of the created PIF objects

Possible Error Codes: `VLAN_TAG_INVALID`

RPC name: create_VLAN_from_PIF *Overview:*

Create a pool-wide VLAN by taking the PIF.

Signature:

```
PIF ref set create_VLAN_from_PIF (session ref session_id, PIF ref pif,
    network ref network, int VLAN)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	pif	physical interface on any particular host, that identifies the PIF on which to create the (pool-wide) VLAN interface
`network ref`	network	network to which this interface should be connected
`int`	VLAN	VLAN tag for the new interface

Minimum Role: pool-operator

Return Type: `PIF ref set`

The references of the created PIF objects

Possible Error Codes: `VLAN_TAG_INVALID`

RPC name: crl_install *Overview:*

Install a TLS CA-issued Certificate Revocation List, pool-wide.

Signature:

```
void crl_install (session ref session_id, string name, string cert)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	A name to give the CRL
`string`	cert	The CRL

Minimum Role: pool-operator

Return Type: `void`

RPC name: crl_list *Overview:*

List the names of all installed TLS CA-issued Certificate Revocation Lists.

Signature:

```
string set crl_list (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `string set`

The names of all installed CRLs

RPC name: crl_uninstall *Overview:*

Remove a pool-wide TLS CA-issued Certificate Revocation List.

Signature:

```
void crl_uninstall (session ref session_id, string name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	The CRL name

Minimum Role: pool-operator

Return Type: `void`

RPC name: deconfigure_wlb *Overview:*

Permanently deconfigures workload balancing monitoring on this pool

Signature:

```
void deconfigure_wlb (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: designate_new_master *Overview:*

Perform an orderly handover of the role of master to the referenced host.

Signature:

```
void designate_new_master (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host who should become the new master

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: detect_nonhomogeneous_external_auth *Overview:*

This call asynchronously detects if the external authentication configuration in any slave is different from that in the master and raises appropriate alerts

Signature:

```
void detect_nonhomogeneous_external_auth (session ref session_id, pool
    ref pool)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	pool	The pool where to detect non-homogeneous external authentication configuration

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: disable_client_certificate_auth *Overview:*

Disable client certificate authentication on the pool

Signature:

```
void disable_client_certificate_auth (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool

Minimum Role: client-cert

Return Type: **`void`**

RPC name: disable_external_auth *Overview:*

This call disables external authentication on all the hosts of the pool

Signature:

```
void disable_external_auth (session ref session_id, pool ref pool, (
    string -> string) map config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	pool	The pool whose external authentication should be disabled
`(string -> string)map`	config	Optional parameters as a list of key-values containing the configuration data

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: disable_ha *Overview:*

Turn off High Availability mode

Signature:

```
void disable_ha (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: client-cert

Return Type: `void`

RPC name: disable_local_storage_caching *Overview:*

This call disables pool-wide local storage caching

Signature:

```
void disable_local_storage_caching (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	Reference to the pool

Minimum Role: pool-operator

Return Type: `void`

RPC name: disable_redo_log *Overview:*

Disable the redo log if in use, unless HA is enabled.

Signature:

```
void disable_redo_log (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: disable_repository_proxy *Overview:*

Disable the proxy for RPM package repositories.

Signature:

```
void disable_repository_proxy (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool

Minimum Role: client-cert

Return Type: `void`

RPC name: disable_ssl_legacy **This message is deprecated.**

Overview:

Sets ssl_legacy false on each host, pool-master last. See Host.ssl_legacy and Host.set_ssl_legacy.

Signature:

```
void disable_ssl_legacy (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	(ignored)

Minimum Role: pool-operator

Return Type: `void`

RPC name: eject *Overview:*

Instruct a pool master to eject a host from the pool

Signature:

```
void eject (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to eject

Minimum Role: pool-operator

Return Type: `void`

RPC name: emergency_reset_master *Overview:*

Instruct a slave already in a pool that the master has changed

Signature:

```
void emergency_reset_master (session ref session_id, string
    master_address)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	master_address	The hostname of the master

Minimum Role: pool-operator

Return Type: `void`

RPC name: emergency_transition_to_master *Overview:*

Instruct host that's currently a slave to transition to being master

Signature:

```
void emergency_transition_to_master (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: enable_client_certificate_auth *Overview:*

Enable client certificate authentication on the pool

Signature:

```
void enable_client_certificate_auth (session ref session_id, pool ref
    self, string name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`string`	name	The name (CN/SAN) that an incoming client certificate must have to allow authentication

Minimum Role: client-cert

Return Type: `void`

RPC name: enable_external_auth *Overview:*

This call enables external authentication on all the hosts of the pool

Signature:

```
void enable_external_auth (session ref session_id, pool ref pool, (
    string -> string) map config, string service_name, string auth_type)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
pool ref	pool	The pool whose external authentication should be enabled
(string -> string)map	config	A list of key-values containing the configuration data
string	service_name	The name of the service
string	auth_type	The type of authentication (e.g. AD for Active Directory)

Minimum Role: pool-admin

Return Type: **void**

RPC name: enable_ha *Overview:*

Turn on High Availability mode

Signature:

```
void enable_ha (session ref session_id, SR ref set heartbeat_srs, (
    string -> string) map configuration)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SR ref set	heartbeat_srs	Set of SRs to use for storage heartbeating
(string -> string)map	configuration	Detailed HA configuration to apply

Minimum Role: client-cert

Return Type: **void**

RPC name: enable_local_storage_caching *Overview:*

This call attempts to enable pool-wide local storage caching

Signature:

```
void enable_local_storage_caching (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	Reference to the pool

Minimum Role: pool-operator

Return Type: `void`

RPC name: enable_redo_log *Overview:*

Enable the redo log on the given SR and start using it, unless HA is enabled.

Signature:

```
void enable_redo_log (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	SR to hold the redo log.

Minimum Role: pool-operator

Return Type: `void`

RPC name: enable_ssl_legacy **This message is removed.**

Overview:

Sets ssl_legacy true on each host, pool-master last. See Host.ssl_legacy and Host.set_ssl_legacy.

Signature:

```
void enable_ssl_legacy (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	(ignored)

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: enable_tls_verification *Overview:*

Enable TLS server certificate verification

Signature:

```
void enable_tls_verification (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the pools known to the system.

Signature:

```
pool ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `pool ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of pool references to pool records for all pools known to the system.

Signature:

```
(pool ref -> pool record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(pool ref -> pool record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given pool.

Signature:

```
pool_allowed_operations set get_allowed_operations (session ref
    session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_allowed_operations set`

value of the field

RPC name: get_blobs *Overview:*

Get the blobs field of the given pool.

Signature:

```
(string -> blob ref) map get_blobs (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> blob ref)map`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the pool instance with the specified UUID.

Signature:

```
pool ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `pool ref`

reference to the object

RPC name: get_client_certificate_auth_enabled *Overview:*

Get the client_certificate_auth_enabled field of the given pool.

Signature:

```
bool get_client_certificate_auth_enabled (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_client_certificate_auth_name *Overview:*

Get the client_certificate_auth_name field of the given pool.

Signature:

```
string get_client_certificate_auth_name (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_coordinator_bias *Overview:*

Get the coordinator_bias field of the given pool.

Signature:

```
bool get_coordinator_bias (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_cpu_info *Overview:*

Get the cpu_info field of the given pool.

Signature:

```
(string -> string) map get_cpu_info (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_crash_dump_SR *Overview:*

Get the crash_dump_SR field of the given pool.

Signature:

```
SR ref get_crash_dump_SR (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given pool.

Signature:

```
(string -> pool_allowed_operations) map get_current_operations (session
    ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> pool_allowed_operations)map`

value of the field

RPC name: get_custom_uefi_certificates *Overview:*

Get the custom_uefi_certificates field of the given pool.

Signature:

```
string get_custom_uefi_certificates (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_default_SR *Overview:*

Get the default_SR field of the given pool.

Signature:

```
SR ref get_default_SR (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_ext_auth_max_threads *Overview:*

Get the ext_auth_max_threads field of the given pool.

Signature:

```
int get_ext_auth_max_threads (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_guest_agent_config *Overview:*

Get the guest_agent_config field of the given pool.

Signature:

```
(string -> string) map get_guest_agent_config (session ref session_id,
    pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_gui_config *Overview:*

Get the gui_config field of the given pool.

Signature:

```
(string -> string) map get_gui_config (session ref session_id, pool ref
     self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_ha_allow_overcommit *Overview:*

Get the ha_allow_overcommit field of the given pool.

Signature:

```
bool get_ha_allow_overcommit (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_ha_cluster_stack *Overview:*

Get the ha_cluster_stack field of the given pool.

Signature:

```
string get_ha_cluster_stack (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_ha_configuration *Overview:*

Get the ha_configuration field of the given pool.

Signature:

```
(string -> string) map get_ha_configuration (session ref session_id,
    pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_ha_enabled *Overview:*

Get the ha_enabled field of the given pool.

Signature:

```
bool get_ha_enabled (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_ha_host_failures_to_tolerate *Overview:*

Get the ha_host_failures_to_tolerate field of the given pool.

Signature:

```
int get_ha_host_failures_to_tolerate (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool ref	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_ha_overcommitted *Overview:*

Get the ha_overcommitted field of the given pool.

Signature:

```
bool get_ha_overcommitted (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool ref	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_ha_plan_exists_for *Overview:*

Get the ha_plan_exists_for field of the given pool.

Signature:

```
int get_ha_plan_exists_for (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_ha_statefiles *Overview:*

Get the ha_statefiles field of the given pool.

Signature:

```
string set get_ha_statefiles (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_health_check_config *Overview:*

Get the health_check_config field of the given pool.

Signature:

```
(string -> string) map get_health_check_config (session ref session_id,
    pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_igmp_snooping_enabled *Overview:*

Get the igmp_snooping_enabled field of the given pool.

Signature:

```
bool get_igmp_snooping_enabled (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_psr_pending *Overview:*

Get the is_psr_pending field of the given pool.

Signature:

```
bool get_is_psr_pending (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_last_update_sync *Overview:*

Get the last_update_sync field of the given pool.

Signature:

```
datetime get_last_update_sync (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_license_state *Overview:*

This call returns the license state for the pool

Signature:

```
(string -> string) map get_license_state (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	Reference to the pool

Minimum Role: read-only

Return Type: `(string -> string)map`

The pool's license state

RPC name: get_live_patching_disabled *Overview:*

Get the live_patching_disabled field of the given pool.

Signature:

```
bool get_live_patching_disabled (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_local_auth_max_threads *Overview:*

Get the local_auth_max_threads field of the given pool.

Signature:

```
int get_local_auth_max_threads (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_master *Overview:*

Get the master field of the given pool.

Signature:

```
host ref get_master (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_metadata_VDIs *Overview:*

Get the metadata_VDIs field of the given pool.

Signature:

```
VDI ref set get_metadata_VDIs (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref set`

value of the field

RPC name: get_migration_compression *Overview:*

Get the migration_compression field of the given pool.

Signature:

```
bool get_migration_compression (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_name_description *Overview:*

Get the name_description field of the given pool.

Signature:

```
string get_name_description (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name_label field of the given pool.

Signature:

```
string get_name_label (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given pool.

Signature:

```
(string -> string) map get_other_config (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_policy_no_vendor_device *Overview:*

Get the policy_no_vendor_device field of the given pool.

Signature:

```
bool get_policy_no_vendor_device (session ref session_id, pool ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given pool.

Signature:

```
pool record get_record (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool record`

all fields from the object

RPC name: get_redo_log_enabled *Overview:*

Get the redo_log_enabled field of the given pool.

Signature:

```
bool get_redo_log_enabled (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_redo_log_vdi *Overview:*

Get the redo_log_vdi field of the given pool.

Signature:

```
VDI ref get_redo_log_vdi (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

RPC name: get_repositories *Overview:*

Get the repositories field of the given pool.

Signature:

```
Repository ref set get_repositories (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Repository ref set`

value of the field

RPC name: get_repository_proxy_password *Overview:*

Get the repository_proxy_password field of the given pool.

Signature:

```
secret ref get_repository_proxy_password (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `secret ref`

value of the field

RPC name: get_repository_proxy_url *Overview:*

Get the repository_proxy_url field of the given pool.

Signature:

```
string get_repository_proxy_url (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_repository_proxy_username *Overview:*

Get the repository_proxy_username field of the given pool.

Signature:

```
string get_repository_proxy_username (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_restrictions *Overview:*

Get the restrictions field of the given pool.

Signature:

```
(string -> string) map get_restrictions (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_suspend_image_SR *Overview:*

Get the suspend_image_SR field of the given pool.

Signature:

```
SR ref get_suspend_image_SR (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_tags *Overview:*

Get the tags field of the given pool.

Signature:

```
string set get_tags (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_telemetry_frequency *Overview:*

Get the telemetry_frequency field of the given pool.

Signature:

```
telemetry_frequency get_telemetry_frequency (session ref session_id,
    pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `telemetry_frequency`

value of the field

RPC name: get_telemetry_next_collection *Overview:*

Get the telemetry_next_collection field of the given pool.

Signature:

```
datetime get_telemetry_next_collection (session ref session_id, pool
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_telemetry_uuid *Overview:*

Get the telemetry_uuid field of the given pool.

Signature:

```
secret ref get_telemetry_uuid (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `secret ref`

value of the field

RPC name: get_tls_verification_enabled *Overview:*

Get the tls_verification_enabled field of the given pool.

Signature:

```
bool get_tls_verification_enabled (session ref session_id, pool ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_uefi_certificates *Overview:*

Get the uefi_certificates field of the given pool.

Signature:

```
string get_uefi_certificates (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_update_sync_day *Overview:*

Get the update_sync_day field of the given pool.

Signature:

```
int get_update_sync_day (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_update_sync_enabled *Overview:*

Get the update_sync_enabled field of the given pool.

Signature:

```
bool get_update_sync_enabled (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_update_sync_frequency *Overview:*

Get the update_sync_frequency field of the given pool.

Signature:

```
update_sync_frequency get_update_sync_frequency (session ref session_id
    , pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `update_sync_frequency`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given pool.

Signature:

```
string get_uuid (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vswitch_controller **This message is deprecated.**

Overview:

Get the vswitch_controller field of the given pool.

Signature:

```
string get_vswitch_controller (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_wlb_enabled *Overview:*

Get the wlb_enabled field of the given pool.

Signature:

```
bool get_wlb_enabled (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_wlb_url *Overview:*

Get the wlb_url field of the given pool.

Signature:

```
string get_wlb_url (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_wlb_username *Overview:*

Get the wlb_username field of the given pool.

Signature:

```
string get_wlb_username (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_wlb_verify_cert **This message is deprecated.**

Overview:

Get the wlb_verify_cert field of the given pool.

Signature:

```
bool get_wlb_verify_cert (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: ha_compute_hypothetical_max_host_failures_to_tolerate *Overview:*

Returns the maximum number of host failures we could tolerate before we would be unable to restart the provided VMs

Signature:

```
int ha_compute_hypothetical_max_host_failures_to_tolerate (session ref
    session_id, (VM ref -> string) map configuration)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`(VM ref -> string)map`	configuration	Map of protected VM reference to restart priority

Minimum Role: read-only

Return Type: `int`

maximum value for ha_host_failures_to_tolerate given provided configuration

RPC name: ha_compute_max_host_failures_to_tolerate *Overview:*

Returns the maximum number of host failures we could tolerate before we would be unable to restart configured VMs

Signature:

```
int ha_compute_max_host_failures_to_tolerate (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `int`

maximum value for ha_host_failures_to_tolerate given current configuration

RPC name: ha_compute_vm_failover_plan *Overview:*

Return a VM failover plan assuming a given subset of hosts fail

Signature:

```
(VM ref -> (string -> string) map) map ha_compute_vm_failover_plan (
    session ref session_id, host ref set failed_hosts, VM ref set
    failed_vms)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref set`	failed_hosts	The set of hosts to assume have failed
`VM ref set`	failed_vms	The set of VMs to restart

Minimum Role: pool-operator

Return Type: `(VM ref -> (string -> string)map)map`

VM failover plan: a map of VM to host to restart the host on

RPC name: ha_failover_plan_exists *Overview:*

Returns true if a VM failover plan exists for up to 'n'host failures

Signature:

```
bool ha_failover_plan_exists (session ref session_id, int n)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`int`	n	The number of host failures to plan for

Minimum Role: pool-operator

Return Type: `bool`

true if a failover plan exists for the supplied number of host failures

RPC name: ha_prevent_restarts_for *Overview:*

When this call returns the VM restart logic will not run for the requested number of seconds. If the argument is zero then the restart thread is immediately unblocked

Signature:

```
void ha_prevent_restarts_for (session ref session_id, int seconds)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`int`	seconds	The number of seconds to block the restart thread for

Minimum Role: pool-operator

Return Type: `void`

RPC name: has_extension *Overview:*

Return true if the extension is available on the pool

Signature:

```
bool has_extension (session ref session_id, pool ref self, string name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`string`	name	The name of the API call

Minimum Role: pool-admin

Return Type: `bool`

True if the extension exists, false otherwise

RPC name: initialize_wlb *Overview:*

Initializes workload balancing monitoring on this pool with the specified wlb server

Signature:

```
void initialize_wlb (session ref session_id, string wlb_url, string
    wlb_username, string wlb_password, string xenserver_username, string
     xenserver_password)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	wlb_url	The ip address and port to use when accessing the wlb server
`string`	wlb_username	The username used to authenticate with the wlb server
`string`	wlb_password	The password used to authenticate with the wlb server
`string`	xenserver_username	The username used by the wlb server to authenticate with the xenserver
`string`	xenserver_password	The password used by the wlb server to authenticate with the xenserver

Minimum Role: pool-operator

Return Type: `void`

RPC name: install_ca_certificate *Overview:*

Install a TLS CA certificate, pool-wide.

Signature:

```
void install_ca_certificate (session ref session_id, string name,
    string cert)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	A name to give the certificate
`string`	cert	The certificate in PEM format

Minimum Role: client-cert

Return Type: `void`

RPC name: join *Overview:*

Instruct host to join a new pool

Signature:

```
void join (session ref session_id, string master_address, string
    master_username, string master_password)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	master_address	The hostname of the master of the pool to join
`string`	master_username	The username of the master (for initial authentication)
`string`	master_password	The password for the master (for initial authentication)

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `JOINING_HOST_CANNOT_CONTAIN_SHARED_SRS`

RPC name: join_force *Overview:*

Instruct host to join a new pool

Signature:

```
void join_force (session ref session_id, string master_address, string
    master_username, string master_password)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	master_address	The hostname of the master of the pool to join
`string`	master_username	The username of the master (for initial authentication)
`string`	master_password	The password for the master (for initial authentication)

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: management_reconfigure *Overview:*

Reconfigure the management network interface for all Hosts in the Pool

Signature:

```
void management_reconfigure (session ref session_id, network ref
    network)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`network ref`	network	The network

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `HA_IS_ENABLED`, `PIF_NOT_PRESENT`, `CANNOT_PLUG_BOND_SLAVE`, `PIF_INCOMPATIBLE_PRIMARY_ADDRESS_TYPE`, `PIF_HAS_NO_NETWORK_CONFIGURATION`, `PIF_HAS_NO_V6_NETWORK_CONFIGURATION`

RPC name: recover_slaves *Overview:*

Instruct a pool master, M, to try and contact its slaves and, if slaves are in emergency mode, reset their master address to M.

Signature:

```
host ref set recover_slaves (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `host ref set`

list of hosts whose master address were successfully reset

RPC name: remove_from_guest_agent_config *Overview:*

Remove a key-value pair from the pool-wide guest agent configuration

Signature:

```
void remove_from_guest_agent_config (session ref session_id, pool ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`string`	key	The key to remove

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: remove_from_gui_config *Overview:*

Remove the given key and its corresponding value from the gui_config field of the given pool. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_gui_config (session ref session_id, pool ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: remove_from_health_check_config *Overview:*

Remove the given key and its corresponding value from the health_check_config field of the given pool. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_health_check_config (session ref session_id, pool ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given pool. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, pool ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_repository *Overview:*

Remove a repository from the enabled set

Signature:

```
void remove_repository (session ref session_id, pool ref self,
    Repository ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`Repository ref`	value	The repository to be removed

Minimum Role: client-cert

Return Type: **`void`**

RPC name: remove_tags *Overview:*

Remove the given value from the tags field of the given pool. If the value is not in that Set, then do nothing.

Signature:

```
void remove_tags (session ref session_id, pool ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	value	Value to remove

Minimum Role: vm-operator

Return Type: `void`

RPC name: reset_telemetry_uuid *Overview:*

Assign a new UUID to telemetry data.

Signature:

```
void reset_telemetry_uuid (session ref session_id, pool ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool

Minimum Role: pool-admin

Return Type: `void`

RPC name: retrieve_wlb_configuration *Overview:*

Retrieves the pool optimization criteria from the workload balancing server

Signature:

```
(string -> string) map retrieve_wlb_configuration (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(string -> string)map`

The configuration used in optimizing this pool

RPC name: retrieve_wlb_recommendations *Overview:*

Retrieves vm migrate recommendations for the pool from the workload balancing server

Signature:

```
(VM ref -> string set) map retrieve_wlb_recommendations (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VM ref -> string set)map`

The list of vm migration recommendations

RPC name: rotate_secret *Overview:*

Signature:

```
void rotate_secret (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-admin

Return Type: `void`

Possible Error Codes: `INTERNAL_ERROR`, `HOST_IS_SLAVE`, `CANNOT_CONTACT_HOST`, `HA_IS_ENABLED`, `NOT_SUPPORTED_DURING_UPGRADE`

RPC name: send_test_post *Overview:*

Send the given body to the given host and port, using HTTPS, and print the response. This is used for debugging the SSL layer.

Signature:

```
string send_test_post (session ref session_id, string host, int port,
    string body)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	host	
`int`	port	
`string`	body	

Minimum Role: pool-admin

Return Type: `string`

The response

RPC name: send_wlb_configuration *Overview:*

Sets the pool optimization criteria for the workload balancing server

Signature:

```
void send_wlb_configuration (session ref session_id, (string -> string)
    map config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`(string -> string)map`	config	The configuration to use in optimizing this pool

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_coordinator_bias *Overview:*

Set the coordinator_bias field of the given pool.

Signature:

```
void set_coordinator_bias (session ref session_id, pool ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool ref	self	reference to the object
bool	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_crash_dump_SR *Overview:*

Set the crash_dump_SR field of the given pool.

Signature:

```
void set_crash_dump_SR (session ref session_id, pool ref self, SR ref
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool ref	self	reference to the object
SR ref	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_custom_uefi_certificates *Overview:*

Set custom UEFI certificates for a pool and all its hosts. Need `allow&#45;custom&#45;uefi&#45;certs` set to true in conf. If empty: default back to Pool.uefi_certificates

Signature:

```
void set_custom_uefi_certificates (session ref session_id, pool ref
    self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`string`	value	The certificates to apply to the pool and its hosts

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_default_SR *Overview:*

Set the default_SR field of the given pool.

Signature:

```
void set_default_SR (session ref session_id, pool ref self, SR ref
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`SR ref`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_ext_auth_max_threads *Overview:*

Signature:

```
void set_ext_auth_max_threads (session ref session_id, pool ref self,
    int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`int`	value	The new maximum

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_gui_config *Overview:*

Set the gui_config field of the given pool.

Signature:

```
void set_gui_config (session ref session_id, pool ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-operator

Return Type: `void`

RPC name: set_ha_allow_overcommit *Overview:*

Set the ha_allow_overcommit field of the given pool.

Signature:

```
void set_ha_allow_overcommit (session ref session_id, pool ref self,
    bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_ha_host_failures_to_tolerate *Overview:*

Set the maximum number of host failures to consider in the HA VM restart planner

Signature:

```
void set_ha_host_failures_to_tolerate (session ref session_id, pool ref
    self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`int`	value	New number of host failures to consider

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_health_check_config *Overview:*

Set the health_check_config field of the given pool.

Signature:

```
void set_health_check_config (session ref session_id, pool ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_https_only *Overview:*

updates all the host firewalls in the pool to open or close port 80 depending on the value

Signature:

```
void set_https_only (session ref session_id, pool ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`bool`	value	true - http port 80 will be blocked, false - http port 80 will be open for the hosts in the pool

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_igmp_snooping_enabled *Overview:*

Enable or disable IGMP Snooping on the pool.

Signature:

```
void set_igmp_snooping_enabled (session ref session_id, pool ref self,
    bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`bool`	value	Enable or disable IGMP Snooping on the pool

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_is_psr_pending *Overview:*

Set the is_psr_pending field of the given pool.

Signature:

```
void set_is_psr_pending (session ref session_id, pool ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_live_patching_disabled *Overview:*

Set the live_patching_disabled field of the given pool.

Signature:

```
void set_live_patching_disabled (session ref session_id, pool ref self,
    bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_local_auth_max_threads *Overview:*

Signature:

```
void set_local_auth_max_threads (session ref session_id, pool ref self,
    int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`int`	value	The new maximum

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_migration_compression *Overview:*

Set the migration_compression field of the given pool.

Signature:

```
void set_migration_compression (session ref session_id, pool ref self,
    bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_description *Overview:*

Set the name_description field of the given pool.

Signature:

```
void set_name_description (session ref session_id, pool ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name_label field of the given pool.

Signature:

```
void set_name_label (session ref session_id, pool ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given pool.

Signature:

```
void set_other_config (session ref session_id, pool ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_policy_no_vendor_device *Overview:*

Set the policy_no_vendor_device field of the given pool.

Signature:

```
void set_policy_no_vendor_device (session ref session_id, pool ref self
    , bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_repositories *Overview:*

Set enabled set of repositories

Signature:

```
void set_repositories (session ref session_id, pool ref self,
    Repository ref set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`Repository ref set`	value	The set of repositories to be enabled

Minimum Role: client-cert

Return Type: **`void`**

RPC name: set_suspend_image_SR *Overview:*

Set the suspend_image_SR field of the given pool.

Signature:

```
void set_suspend_image_SR (session ref session_id, pool ref self, SR
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`SR ref`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_tags *Overview:*

Set the tags field of the given pool.

Signature:

```
void set_tags (session ref session_id, pool ref self, string set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`string set`	value	New value to set

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: set_telemetry_next_collection *Overview:*

Set the timestamp for the next telemetry data collection.

Signature:

```
void set_telemetry_next_collection (session ref session_id, pool ref
    self, datetime value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`datetime`	value	The earliest timestamp (in UTC) when the next round of telemetry collection can be carried out.

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_uefi_certificates   This message is deprecated.

Overview:

Set the UEFI certificates for a pool and all its hosts. Deprecated: use set_custom_uefi_certificates instead

Signature:

```
void set_uefi_certificates (session ref session_id, pool ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`string`	value	The certificates to apply to the pool and its hosts

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_update_sync_enabled *Overview:*

enable or disable periodic update synchronization depending on the value

Signature:

```
void set_update_sync_enabled (session ref session_id, pool ref self,
    bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`bool`	value	true - enable periodic update synchronization, false - disable it

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_vswitch_controller **This message is deprecated.**

Overview:

Set the IP address of the vswitch controller.

Signature:

```
void set_vswitch_controller (session ref session_id, string address)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	address	IP address of the vswitch controller.

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_wlb_enabled *Overview:*

Set the wlb_enabled field of the given pool.

Signature:

```
void set_wlb_enabled (session ref session_id, pool ref self, bool value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: client-cert

Return Type: **`void`**

RPC name: set_wlb_verify_cert **This message is deprecated.**

Overview:

Set the wlb_verify_cert field of the given pool.

Signature:

```
void set_wlb_verify_cert (session ref session_id, pool ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: sync_database *Overview:*

Forcibly synchronise the database now

Signature:

```
void sync_database (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `void`

RPC name: sync_updates *Overview:*

Sync with the enabled repository

Signature:

```
string sync_updates (session ref session_id, pool ref self, bool force,
     string token, string token_id)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	The pool
`bool`	force	If true local mirroring repo will be removed before syncing
`string`	token	The token for repository client authentication
`string`	token_id	The ID of the token

Minimum Role: client-cert

Return Type: `string`

The SHA256 hash of updateinfo.xml.gz

RPC name: test_archive_target *Overview:*

This call tests if a location is valid

Signature:

```
string test_archive_target (session ref session_id, pool ref self, (
    string -> string) map config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool ref`	self	Reference to the pool
`(string -> string)map`	config	Location config settings to test

Minimum Role: pool-operator

Return Type: `string`

An XMLRPC result

RPC name: uninstall_ca_certificate *Overview:*

Remove a pool-wide TLS CA certificate.

Signature:

```
void uninstall_ca_certificate (session ref session_id, string name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name	The certificate name

Minimum Role: client-cert

Return Type: **`void`**

Class: pool_patch

This class is deprecated.

Pool-wide patches

Fields for class: pool_patch

Field	Type	Qualifier	Description
after_apply_guidance	`after_apply_guidance set`	*RO/runtime*	**Deprecated**. What the client should do after this patch has been applied.
host_patches	`host_patch ref set`	*RO/runtime*	**Deprecated**. This hosts this patch is applied to.
name_description	`string`	*RO/constructor*	**Deprecated**. a notes field containing human-readable description
name_label	`string`	*RO/constructor*	**Deprecated**. a human-readable name
other_config	`(string -> string)map`	*RW*	**Deprecated**. additional configuration
pool_applied	`bool`	*RO/runtime*	**Deprecated**. This patch should be applied across the entire pool
pool_update	`pool_update ref`	*RO/constructor*	**Deprecated**. A reference to the associated pool_update object
size	`int`	*RO/runtime*	**Deprecated**. Size of the patch
uuid	`string`	*RO/runtime*	**Deprecated**. Unique identifier/object reference
version	`string`	*RO/constructor*	**Deprecated**. Patch version number

RPCs associated with class: pool_patch

RPC name: add_to_other_config   This message is deprecated.

Overview:

Add the given key-value pair to the other_config field of the given pool_patch.

Signature:

```
void add_to_other_config (session ref session_id, pool_patch ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: apply   This message is deprecated.

Overview:

Apply the selected patch to a host and return its output

Signature:

```
string apply (session ref session_id, pool_patch ref self, host ref
    host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	The patch to apply

type	name	description
host ref	host	The host to apply the patch too

Minimum Role: pool-operator

Return Type: `string`

the output of the patch application process

RPC name: clean **This message is deprecated.**

Overview:

Removes the patch's files from the server

Signature:

```
void clean (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool_patch ref	self	The patch to clean up

Minimum Role: pool-operator

Return Type: `void`

RPC name: clean_on_host **This message is deprecated.**

Overview:

Removes the patch's files from the specified host

Signature:

```
void clean_on_host (session ref session_id, pool_patch ref self, host
    ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	The patch to clean up
`host ref`	host	The host on which to clean the patch

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: destroy   This message is deprecated.

Overview:

Removes the patch's files from all hosts in the pool, and removes the database entries. Only works on unapplied patches.

Signature:

```
void destroy (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	The patch to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_after_apply_guidance   This message is deprecated.

Overview:

Get the after_apply_guidance field of the given pool_patch.

Signature:

```
after_apply_guidance set get_after_apply_guidance (session ref
    session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `after_apply_guidance set`

value of the field

RPC name: get_all   This message is deprecated.

Overview:

Return a list of all the pool_patchs known to the system.

Signature:

```
pool_patch ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `pool_patch ref set`

references to all objects

RPC name: get_all_records   This message is deprecated.

Overview:

Return a map of pool_patch references to pool_patch records for all pool_patchs known to the system.

Signature:

```
(pool_patch ref -> pool_patch record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(pool_patch ref -> pool_patch record)map`

records of all objects

RPC name: get_by_name_label This message is deprecated.

Overview:

Get all the pool_patch instances with the given label.

Signature:

```
pool_patch ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `pool_patch ref set`

references to objects with matching names

RPC name: get_by_uuid This message is deprecated.

Overview:

Get a reference to the pool_patch instance with the specified UUID.

Signature:

```
pool_patch ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `pool_patch ref`

reference to the object

RPC name: get_host_patches **This message is deprecated.**

Overview:

Get the host_patches field of the given pool_patch.

Signature:

```
host_patch ref set get_host_patches (session ref session_id, pool_patch
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host_patch ref set`

value of the field

RPC name: get_name_description **This message is deprecated.**

Overview:

Get the name/description field of the given pool_patch.

Signature:

```
string get_name_description (session ref session_id, pool_patch ref
   self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label **This message is deprecated.**

Overview:

Get the name/label field of the given pool_patch.

Signature:

```
string get_name_label (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config **This message is deprecated.**

Overview:

Get the other_config field of the given pool_patch.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
   pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_pool_applied   This message is deprecated.

Overview:

Get the pool_applied field of the given pool_patch.

Signature:

```
bool get_pool_applied (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_pool_update   This message is deprecated.

Overview:

Get the pool_update field of the given pool_patch.

Signature:

```
pool_update ref get_pool_update (session ref session_id, pool_patch ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_update ref`

value of the field

RPC name: get_record This message is deprecated.

Overview:

Get a record containing the current state of the given pool_patch.

Signature:

```
pool_patch record get_record (session ref session_id, pool_patch ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_patch record`

all fields from the object

RPC name: get_size This message is deprecated.

Overview:

Get the size field of the given pool_patch.

Signature:

```
int get_size (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_uuid **This message is deprecated.**

Overview:

Get the uuid field of the given pool_patch.

Signature:

```
string get_uuid (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_version **This message is deprecated.**

Overview:

Get the version field of the given pool_patch.

Signature:

```
string get_version (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: pool_apply This message is deprecated.

Overview:

Apply the selected patch to all hosts in the pool and return a map of host_ref -> patch output

Signature:

```
void pool_apply (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	The patch to apply

Minimum Role: pool-operator

Return Type: `void`

RPC name: pool_clean This message is deprecated.

Overview:

Removes the patch's files from all hosts in the pool, but does not remove the database entries

Signature:

```
void pool_clean (session ref session_id, pool_patch ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	The patch to clean up

Minimum Role: pool-operator

Return Type: `void`

RPC name: precheck This message is deprecated.

Overview:

Execute the precheck stage of the selected patch on a host and return its output

Signature:

```
string precheck (session ref session_id, pool_patch ref self, host ref
    host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	The patch whose prechecks will be run
`host ref`	host	The host to run the prechecks on

Minimum Role: pool-operator

Return Type: `string`

the output of the patch prechecks

RPC name: remove_from_other_config This message is deprecated.

Overview:

Remove the given key and its corresponding value from the other_config field of the given pool_patch. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, pool_patch ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_patch ref`	self	reference to the object

type	name	description
string	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config This message is deprecated.

Overview:

Set the other_config field of the given pool_patch.

Signature:

```
void set_other_config (session ref session_id, pool_patch ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool_patch ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: pool_update

Pool-wide updates to the host software

Fields for class: pool_update

Field	Type	Qualifier	Description
after_apply_guidance	update_after_apply_guidance set	RO/constructor	What the client should do after this update has been applied.
enforce_homogeneity	bool	RO/constructor	Flag - if true, all hosts in a pool must apply this update
hosts	host ref set	RO/runtime	The hosts that have applied this update.
installation_size	int	RO/constructor	Size of the update in bytes
key	string	RO/constructor	GPG key of the update
name_description	string	RO/constructor	a notes field containing human-readable description
name_label	string	RO/constructor	a human-readable name
other_config	(string -> string)map	RW	additional configuration
uuid	string	RO/runtime	Unique identifier/object reference
vdi	VDI ref	RO/constructor	VDI the update was uploaded to
version	string	RO/constructor	Update version number

RPCs associated with class: pool_update

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given pool_update.

Signature:

```
void add_to_other_config (session ref session_id, pool_update ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool_update ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: pool-operator

Return Type: **void**

RPC name: apply *Overview:*

Apply the selected update to a host

Signature:

```
void apply (session ref session_id, pool_update ref self, host ref host
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool_update ref	self	The update to apply
host ref	host	The host to apply the update to.

Minimum Role: pool-operator

Return Type: **void**

RPC name: destroy *Overview:*

Removes the database entry. Only works on unapplied update.

Signature:

```
void destroy (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	The update to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_after_apply_guidance *Overview:*

Get the after_apply_guidance field of the given pool_update.

Signature:

```
update_after_apply_guidance set get_after_apply_guidance (session ref
    session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `update_after_apply_guidance set`

value of the field

RPC name: get_all *Overview:*

Return a list of all the pool_updates known to the system.

Signature:

```
pool_update ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `pool_update ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of pool_update references to pool_update records for all pool_updates known to the system.

Signature:

```
(pool_update ref -> pool_update record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(pool_update ref -> pool_update record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the pool_update instances with the given label.

Signature:

```
pool_update ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `pool_update ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the pool_update instance with the specified UUID.

Signature:

```
pool_update ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `pool_update ref`

reference to the object

RPC name: get_enforce_homogeneity *Overview:*

Get the enforce_homogeneity field of the given pool_update.

Signature:

```
bool get_enforce_homogeneity (session ref session_id, pool_update ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_hosts *Overview:*

Get the hosts field of the given pool_update.

Signature:

```
host ref set get_hosts (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref set`

value of the field

RPC name: get_installation_size *Overview:*

Get the installation_size field of the given pool_update.

Signature:

```
int get_installation_size (session ref session_id, pool_update ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_key *Overview:*

Get the key field of the given pool_update.

Signature:

```
string get_key (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given pool_update.

Signature:

```
string get_name_description (session ref session_id, pool_update ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given pool_update.

Signature:

```
string get_name_label (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool_update ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given pool_update.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
pool_update ref	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given pool_update.

Signature:

```
pool_update record get_record (session ref session_id, pool_update ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool_update record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given pool_update.

Signature:

```
string get_uuid (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vdi *Overview:*

Get the vdi field of the given pool_update.

Signature:

```
VDI ref get_vdi (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

RPC name: get_version *Overview:*

Get the version field of the given pool_update.

Signature:

```
string get_version (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: introduce *Overview:*

Introduce update VDI

Signature:

```
pool_update ref introduce (session ref session_id, VDI ref vdi)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI which contains a software update.

Minimum Role: pool-operator

Return Type: `pool_update ref`

the introduced pool update

RPC name: pool_apply *Overview:*

Apply the selected update to all hosts in the pool

Signature:

```
void pool_apply (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	The update to apply

Minimum Role: pool-operator

Return Type: `void`

RPC name: pool_clean *Overview:*

Removes the update's files from all hosts in the pool, but does not revert the update

Signature:

```
void pool_clean (session ref session_id, pool_update ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	The update to clean up

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: precheck *Overview:*

Execute the precheck stage of the selected update on a host

Signature:

```
livepatch_status precheck (session ref session_id, pool_update ref self
    , host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	The update whose prechecks will be run
`host ref`	host	The host to run the prechecks on.

Minimum Role: pool-operator

Return Type: `livepatch_status`

The precheck pool update

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given pool_update. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, pool_update ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given pool_update.

Signature:

```
void set_other_config (session ref session_id, pool_update ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`pool_update ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Class: probe_result

A set of properties that describe one result element of SR.probe. Result elements and properties can change dynamically based on changes to the the SR.probe input-parameters or the target.

Fields for class: probe_result

Field	Type	Qualifier	Description
complete	`bool`	*RO/runtime*	True if this configuration is complete and can be used to call SR.create. False if it requires further iterative calls to SR.probe, to potentially narrow down on a configuration that can be used.
configuration	`(string -> string)map`	*RO/runtime*	Plugin-specific configuration which describes where and how to locate the storage repository. This may include the physical block device name, a remote NFS server and path or an RBD storage pool.
extra_info	`(string -> string)map`	*RO/runtime*	Additional plugin-specific information about this configuration, that might be of use for an API user. This can for example include the LUN or the WWPN.
sr	`sr_stat record option`	*RO/runtime*	Existing SR found for this configuration

RPCs associated with class: probe_result

Class probe_result has no additional RPCs associated with it.

Class: PUSB

A physical USB device

Fields for class: PUSB

Field	Type	Qualifier	Description
description	`string`	*RO/constructor*	USB device description
host	`host ref`	*RO/constructor*	Physical machine that owns the USB device
other_config	`(string -> string)map`	*RW*	additional configuration
passthrough_enabled	`bool`	*RO/runtime*	enabled for passthrough
path	`string`	*RO/constructor*	port path of USB device
product_desc	`string`	*RO/constructor*	product description of the USB device
product_id	`string`	*RO/constructor*	product id of the USB device
serial	`string`	*RO/constructor*	serial of the USB device
speed	`float`	*RO/constructor*	USB device speed
USB_group	`USB_group ref`	*RO/constructor*	USB group the PUSB is contained in
uuid	`string`	*RO/runtime*	Unique identifier/object reference
vendor_desc	`string`	*RO/constructor*	vendor description of the USB device
vendor_id	`string`	*RO/constructor*	vendor id of the USB device
version	`string`	*RO/constructor*	USB device version

RPCs associated with class: PUSB

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given PUSB.

Signature:

```
void add_to_other_config (session ref session_id, PUSB ref self, string
     key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the PUSBs known to the system.

Signature:

```
PUSB ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PUSB ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PUSB references to PUSB records for all PUSBs known to the system.

Signature:

```
(PUSB ref -> PUSB record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PUSB ref -> PUSB record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PUSB instance with the specified UUID.

Signature:

```
PUSB ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PUSB ref`

reference to the object

RPC name: get_description *Overview:*

Get the description field of the given PUSB.

Signature:

```
string get_description (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_host *Overview:*

Get the host field of the given PUSB.

Signature:

```
host ref get_host (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given PUSB.

Signature:

```
(string -> string) map get_other_config (session ref session_id, PUSB
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_passthrough_enabled *Overview:*

Get the passthrough_enabled field of the given PUSB.

Signature:

```
bool get_passthrough_enabled (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_path *Overview:*

Get the path field of the given PUSB.

Signature:

```
string get_path (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_product_desc *Overview:*

Get the product_desc field of the given PUSB.

Signature:

```
string get_product_desc (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_product_id *Overview:*

Get the product_id field of the given PUSB.

Signature:

```
string get_product_id (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PUSB.

Signature:

```
PUSB record get_record (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PUSB record`

all fields from the object

RPC name: get_serial *Overview:*

Get the serial field of the given PUSB.

Signature:

```
string get_serial (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_speed *Overview:*

Get the speed field of the given PUSB.

Signature:

```
float get_speed (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PUSB ref	self	reference to the object

Minimum Role: read-only

Return Type: **float**

value of the field

RPC name: get_USB_group *Overview:*

Get the USB_group field of the given PUSB.

Signature:

```
USB_group ref get_USB_group (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PUSB ref	self	reference to the object

Minimum Role: read-only

Return Type: USB_group ref

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PUSB.

Signature:

```
string get_uuid (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor_desc *Overview:*

Get the vendor_desc field of the given PUSB.

Signature:

```
string get_vendor_desc (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor_id *Overview:*

Get the vendor_id field of the given PUSB.

Signature:

```
string get_vendor_id (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_version *Overview:*

Get the version field of the given PUSB.

Signature:

```
string get_version (session ref session_id, PUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given PUSB. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, PUSB ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-admin

Return Type: `void`

RPC name: scan *Overview:*

Signature:

```
void scan (session ref session_id, host ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given PUSB.

Signature:

```
void set_other_config (session ref session_id, PUSB ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_passthrough_enabled *Overview:*

Signature:

```
void set_passthrough_enabled (session ref session_id, PUSB ref self,
    bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PUSB ref`	self	this PUSB
`bool`	value	passthrough is enabled when true and disabled with false

Minimum Role: pool-admin

Return Type: `void`

Class: PVS_cache_storage

Describes the storage that is available to a PVS site for caching purposes

Fields for class: PVS_cache_storage

Field	Type	Qualifier	Description
host	`host ref`	*RO/constructor*	The host on which this object defines PVS cache storage
site	`PVS_site ref`	*RO/constructor*	The PVS_site for which this object defines the storage
size	**`int`**	*RO/constructor*	The size of the cache VDI (in bytes)
SR	`SR ref`	*RO/constructor*	SR providing storage for the PVS cache
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VDI	`VDI ref`	*RO/runtime*	The VDI used for caching

RPCs associated with class: PVS_cache_storage

RPC name: create *Overview:*

Create a new PVS_cache_storage instance, and return its handle.

Signature:

```
PVS_cache_storage ref create (session ref session_id, PVS_cache_storage
    record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
PVS_cache_storage record	args	All constructor arguments

Minimum Role: pool-operator

Return Type: `PVS_cache_storage ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified PVS_cache_storage instance.

Signature:

```
void destroy (session ref session_id, PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_cache_storage ref	self	reference to the object

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the PVS_cache_storages known to the system.

Signature:

```
PVS_cache_storage ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PVS_cache_storage ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PVS_cache_storage references to PVS_cache_storage records for all PVS_cache_storages known to the system.

Signature:

```
(PVS_cache_storage ref -> PVS_cache_storage record) map get_all_records
    (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PVS_cache_storage ref -> PVS_cache_storage record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PVS_cache_storage instance with the specified UUID.

Signature:

```
PVS_cache_storage ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PVS_cache_storage ref`

reference to the object

RPC name: get_host *Overview:*

Get the host field of the given PVS_cache_storage.

Signature:

```
host ref get_host (session ref session_id, PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_cache_storage ref	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PVS_cache_storage.

Signature:

```
PVS_cache_storage record get_record (session ref session_id,
    PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_cache_storage ref	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_cache_storage record`

all fields from the object

RPC name: get_site *Overview:*

Get the site field of the given PVS_cache_storage.

Signature:

```
PVS_site ref get_site (session ref session_id, PVS_cache_storage ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_cache_storage ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_site ref`

value of the field

RPC name: get_size *Overview:*

Get the size field of the given PVS_cache_storage.

Signature:

```
int get_size (session ref session_id, PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_cache_storage ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_SR *Overview:*

Get the SR field of the given PVS_cache_storage.

Signature:

```
SR ref get_SR (session ref session_id, PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_cache_storage ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PVS_cache_storage.

Signature:

```
string get_uuid (session ref session_id, PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_cache_storage ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VDI *Overview:*

Get the VDI field of the given PVS_cache_storage.

Signature:

```
VDI ref get_VDI (session ref session_id, PVS_cache_storage ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_cache_storage ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

Class: PVS_proxy

a proxy connects a VM/VIF with a PVS site

Fields for class: PVS_proxy

Field	Type	Qualifier	Description
currently_attached	`bool`	*RO/runtime*	true = VM is currently proxied
site	`PVS_site ref`	*RO/constructor*	PVS site this proxy is part of
status	`pvs_proxy_status`	*RO/runtime*	The run-time status of the proxy
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VIF	`VIF ref`	*RO/constructor*	VIF of the VM using the proxy

RPCs associated with class: PVS_proxy

RPC name: create *Overview:*

Configure a VM/VIF to use a PVS proxy

Signature:

```
PVS_proxy ref create (session ref session_id, PVS_site ref site, VIF
    ref VIF)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	site	PVS site that we proxy for
`VIF ref`	VIF	VIF for the VM that needs to be proxied

Minimum Role: pool-operator

Return Type: `PVS_proxy ref`

the new PVS proxy

RPC name: destroy *Overview:*

remove (or switch off) a PVS proxy for this VM

Signature:

```
void destroy (session ref session_id, PVS_proxy ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	this PVS proxy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the PVS_proxys known to the system.

Signature:

```
PVS_proxy ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PVS_proxy ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PVS_proxy references to PVS_proxy records for all PVS_proxys known to the system.

Signature:

```
(PVS_proxy ref -> PVS_proxy record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PVS_proxy ref -> PVS_proxy record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PVS_proxy instance with the specified UUID.

Signature:

```
PVS_proxy ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PVS_proxy ref`

reference to the object

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given PVS_proxy.

Signature:

```
bool get_currently_attached (session ref session_id, PVS_proxy ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PVS_proxy.

Signature:

```
PVS_proxy record get_record (session ref session_id, PVS_proxy ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_proxy record`

all fields from the object

RPC name: get_site *Overview:*

Get the site field of the given PVS_proxy.

Signature:

```
PVS_site ref get_site (session ref session_id, PVS_proxy ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_site ref`

value of the field

RPC name: get_status *Overview:*

Get the status field of the given PVS_proxy.

Signature:

```
pvs_proxy_status get_status (session ref session_id, PVS_proxy ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pvs_proxy_status`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PVS_proxy.

Signature:

```
string get_uuid (session ref session_id, PVS_proxy ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VIF *Overview:*

Get the VIF field of the given PVS_proxy.

Signature:

```
VIF ref get_VIF (session ref session_id, PVS_proxy ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_proxy ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VIF ref`

value of the field

Class: PVS_server

individual machine serving provisioning (block) data

Fields for class: PVS_server

Field	Type	Qualifier	Description
addresses	`string set`	*RO/constructor*	IPv4 addresses of this server
first_port	`int`	*RO/constructor*	First UDP port accepted by this server
last_port	`int`	*RO/constructor*	Last UDP port accepted by this server
site	`PVS_site ref`	*RO/constructor*	PVS site this server is part of
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: PVS_server

RPC name: forget *Overview:*

forget a PVS server

Signature:

```
void forget (session ref session_id, PVS_server ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	this PVS server

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_addresses *Overview:*

Get the addresses field of the given PVS_server.

Signature:

```
string set get_addresses (session ref session_id, PVS_server ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_all *Overview:*

Return a list of all the PVS_servers known to the system.

Signature:

```
PVS_server ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PVS_server ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PVS_server references to PVS_server records for all PVS_servers known to the system.

Signature:

```
(PVS_server ref -> PVS_server record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PVS_server ref -> PVS_server record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the PVS_server instance with the specified UUID.

Signature:

```
PVS_server ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PVS_server ref`

reference to the object

RPC name: get_first_port *Overview:*

Get the first_port field of the given PVS_server.

Signature:

```
int get_first_port (session ref session_id, PVS_server ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_last_port *Overview:*

Get the last_port field of the given PVS_server.

Signature:

```
int get_last_port (session ref session_id, PVS_server ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PVS_server.

Signature:

```
PVS_server record get_record (session ref session_id, PVS_server ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_server record`

all fields from the object

RPC name: get_site *Overview:*

Get the site field of the given PVS_server.

Signature:

```
PVS_site ref get_site (session ref session_id, PVS_server ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_site ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PVS_server.

Signature:

```
string get_uuid (session ref session_id, PVS_server ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_server ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: introduce *Overview:*

introduce new PVS server

Signature:

```
PVS_server ref introduce (session ref session_id, string set addresses,
    int first_port, int last_port, PVS_site ref site)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string set`	addresses	IPv4/IPv6 addresses of the server
`int`	first_port	first UDP port accepted by this server
`int`	last_port	last UDP port accepted by this server
`PVS_site ref`	site	PVS site this server is a part of

Minimum Role: pool-operator

Return Type: `PVS_server ref`

the new PVS server

Class: PVS_site

machines serving blocks of data for provisioning VMs

Fields for class: PVS_site

Field	Type	Qualifier	Description
cache_storage	`PVS_cache_storage ref set`	*RO/runtime*	The SR used by PVS proxy for the cache

Field	Type	Qualifier	Description
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
proxies	`PVS_proxy ref set`	*RO/runtime*	The set of proxies associated with the site
PVS_uuid	`string`	*RO/constructor*	Unique identifier of the PVS site, as configured in PVS
servers	`PVS_server ref set`	*RO/runtime*	The set of PVS servers in the site
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: PVS_site

RPC name: forget *Overview:*

Remove a site's meta data

Signature:

```
void forget (session ref session_id, PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	this PVS site

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `PVS_SITE_CONTAINS_RUNNING_PROXIES`, `PVS_SITE_CONTAINS_SERVERS`

RPC name: get_all *Overview:*

Return a list of all the PVS_sites known to the system.

Signature:

```
PVS_site ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `PVS_site ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of PVS_site references to PVS_site records for all PVS_sites known to the system.

Signature:

```
(PVS_site ref -> PVS_site record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(PVS_site ref -> PVS_site record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the PVS_site instances with the given label.

Signature:

```
PVS_site ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `PVS_site ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the PVS_site instance with the specified UUID.

Signature:

```
PVS_site ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `PVS_site ref`

reference to the object

RPC name: get_cache_storage *Overview:*

Get the cache_storage field of the given PVS_site.

Signature:

```
PVS_cache_storage ref set get_cache_storage (session ref session_id,
    PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_cache_storage ref set`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given PVS_site.

Signature:

```
string get_name_description (session ref session_id, PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given PVS_site.

Signature:

```
string get_name_label (session ref session_id, PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_proxies *Overview:*

Get the proxies field of the given PVS_site.

Signature:

```
PVS_proxy ref set get_proxies (session ref session_id, PVS_site ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_proxy ref set`

value of the field

RPC name: get_PVS_uuid *Overview:*

Get the PVS_uuid field of the given PVS_site.

Signature:

```
string get_PVS_uuid (session ref session_id, PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given PVS_site.

Signature:

```
PVS_site record get_record (session ref session_id, PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PVS_site record`

all fields from the object

RPC name: get_servers *Overview:*

Get the servers field of the given PVS_site.

Signature:

```
PVS_server ref set get_servers (session ref session_id, PVS_site ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_site ref	self	reference to the object

Minimum Role: read-only

Return Type: PVS_server ref set

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given PVS_site.

Signature:

```
string get_uuid (session ref session_id, PVS_site ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_site ref	self	reference to the object

Minimum Role: read-only

Return Type: string

value of the field

RPC name: introduce *Overview:*

Introduce new PVS site

Signature:

```
PVS_site ref introduce (session ref session_id, string name_label,
    string name_description, string PVS_uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name_label	name of the PVS site
`string`	name_description	description of the PVS site
`string`	PVS_uuid	unique identifier of the PVS site

Minimum Role: pool-operator

Return Type: `PVS_site ref`

the new PVS site

RPC name: set_name_description *Overview:*

Set the name/description field of the given PVS_site.

Signature:

```
void set_name_description (session ref session_id, PVS_site ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PVS_site ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name/label field of the given PVS_site.

Signature:

```
void set_name_label (session ref session_id, PVS_site ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_site ref	self	reference to the object
string	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_PVS_uuid *Overview:*

Update the PVS UUID of the PVS site

Signature:

```
void set_PVS_uuid (session ref session_id, PVS_site ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
PVS_site ref	self	this PVS site
string	value	PVS UUID to be used

Minimum Role: pool-operator

Return Type: **void**

Class: Repository

Repository for updates

Fields for class: Repository

Field	Type	Qualifier	Description
binary_url	`string`	*RO/constructor*	Base URL of binary packages in this repository
gpgkey_path	`string`	*RO/constructor*	The file name of the GPG public key of this repository
hash	`string`	*RO/runtime*	SHA256 checksum of latest updateinfo.xml.gz in this repository if its 'update'is true
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
source_url	`string`	*RO/constructor*	Base URL of source packages in this repository
up_to_date	`bool`	*RO/runtime*	**Removed**. True if all hosts in pool is up to date with this repository
update	`bool`	*RO/constructor*	True if updateinfo.xml in this repository needs to be parsed
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: Repository

RPC name: forget *Overview:*

Remove the repository record from the database

Signature:

```
void forget (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	The repository to be removed from the database

Minimum Role: client-cert

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the Repositorys known to the system.

Signature:

```
Repository ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `Repository ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of Repository references to Repository records for all Repositorys known to the system.

Signature:

```
(Repository ref -> Repository record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(Repository ref -> Repository record)map`

records of all objects

RPC name: get_binary_url *Overview:*

Get the binary_url field of the given Repository.

Signature:

```
string get_binary_url (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the Repository instances with the given label.

Signature:

```
Repository ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `Repository ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the Repository instance with the specified UUID.

Signature:

```
Repository ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `Repository ref`

reference to the object

RPC name: get_gpgkey_path *Overview:*

Get the gpgkey_path field of the given Repository.

Signature:

```
string get_gpgkey_path (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_hash *Overview:*

Get the hash field of the given Repository.

Signature:

```
string get_hash (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given Repository.

Signature:

```
string get_name_description (session ref session_id, Repository ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given Repository.

Signature:

```
string get_name_label (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given Repository.

Signature:

```
Repository record get_record (session ref session_id, Repository ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `Repository record`

all fields from the object

RPC name: get_source_url *Overview:*

Get the source_url field of the given Repository.

Signature:

```
string get_source_url (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_up_to_date **This message is removed.**

Overview:

Get the up_to_date field of the given Repository.

Signature:

```
bool get_up_to_date (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_update *Overview:*

Get the update field of the given Repository.

Signature:

```
bool get_update (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given Repository.

Signature:

```
string get_uuid (session ref session_id, Repository ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: introduce *Overview:*

Add the configuration for a new repository

Signature:

```
Repository ref introduce (session ref session_id, string name_label,
    string name_description, string binary_url, string source_url, bool
    update, string gpgkey_path)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	name_label	The name of the repository
`string`	name_description	The description of the repository
`string`	binary_url	Base URL of binary packages in this repository
`string`	source_url	Base URL of source packages in this repository
`bool`	update	True if the repository is an update repository. This means that updateinfo.xml will be parsed
`string`	gpgkey_path	The GPG public key file name

Minimum Role: client-cert

Return Type: `Repository ref`

The ref of the created repository record.

RPC name: set_gpgkey_path *Overview:*

Set the file name of the GPG public key of the repository

Signature:

```
void set_gpgkey_path (session ref session_id, Repository ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
Repository ref	self	The repository
string	value	The file name of the GPG public key of the repository

Minimum Role: client-cert

Return Type: `void`

RPC name: set_name_description *Overview:*

Set the name/description field of the given Repository.

Signature:

```
void set_name_description (session ref session_id, Repository ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
Repository ref	self	reference to the object
string	value	New value to set

Minimum Role: client-cert

Return Type: `void`

RPC name: set_name_label *Overview:*

Set the name/label field of the given Repository.

Signature:

```
void set_name_label (session ref session_id, Repository ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`Repository ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: client-cert

Return Type: **`void`**

Class: role

A set of permissions associated with a subject

Fields for class: role

Field	Type	Qualifier	Description
is_internal	`bool`	*RO/runtime*	Indicates whether the role is only to be assigned internally by xapi, or can be used by clients
name_description	`string`	*RO/constructor*	what this role is for
name_label	`string`	*RO/constructor*	a short user-friendly name for the role
subroles	`role ref set`	*RO/constructor*	a list of pointers to other roles or permissions
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: role

RPC name: get_all *Overview:*

Return a list of all the roles known to the system.

Signature:

```
role ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `role ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of role references to role records for all roles known to the system.

Signature:

```
(role ref -> role record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(role ref -> role record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the role instances with the given label.

Signature:

```
role ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `role ref set`

references to objects with matching names

RPC name: get_by_permission *Overview:*

This call returns a list of roles given a permission

Signature:

```
role ref set get_by_permission (session ref session_id, role ref
    permission)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	permission	a reference to a permission

Minimum Role: read-only

Return Type: `role ref set`

a list of references to roles

RPC name: get_by_permission_name_label *Overview:*

This call returns a list of roles given a permission name

Signature:

```
role ref set get_by_permission_name_label (session ref session_id,
    string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	The short friendly name of the role

Minimum Role: read-only

Return Type: `role ref set`

a list of references to roles

RPC name: get_by_uuid *Overview:*

Get a reference to the role instance with the specified UUID.

Signature:

```
role ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `role ref`

reference to the object

RPC name: get_is_internal *Overview:*

Get the is_internal field of the given role.

Signature:

```
bool get_is_internal (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given role.

Signature:

```
string get_name_description (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given role.

Signature:

```
string get_name_label (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_permissions *Overview:*

This call returns a list of permissions given a role

Signature:

```
role ref set get_permissions (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	a reference to a role

Minimum Role: read-only

Return Type: `role ref set`

a list of permissions

RPC name: get_permissions_name_label *Overview:*

This call returns a list of permission names given a role

Signature:

```
string set get_permissions_name_label (session ref session_id, role ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	a reference to a role

Minimum Role: read-only

Return Type: `string set`

a list of permission names

RPC name: get_record *Overview:*

Get a record containing the current state of the given role.

Signature:

```
role record get_record (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	reference to the object

Minimum Role: read-only

Return Type: `role record`

all fields from the object

RPC name: get_subroles *Overview:*

Get the subroles field of the given role.

Signature:

```
role ref set get_subroles (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`role ref`	self	reference to the object

Minimum Role: read-only

Return Type: `role ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given role.

Signature:

```
string get_uuid (session ref session_id, role ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
role ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

Class: SDN_controller

Describes the SDN controller that is to connect with the pool

Fields for class: SDN_controller

Field	Type	Qualifier	Description
address	`string`	*RO/constructor*	IP address of the controller
port	`int`	*RO/constructor*	TCP port of the controller
protocol	`sdn_controller_protocol`	*RO/constructor*	Protocol to connect with SDN controller
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: SDN_controller

RPC name: forget *Overview:*

Remove the OVS manager of the pool and destroy the db record.

Signature:

```
void forget (session ref session_id, SDN_controller ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SDN_controller ref`	self	this SDN controller

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_address *Overview:*

Get the address field of the given SDN_controller.

Signature:

```
string get_address (session ref session_id, SDN_controller ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SDN_controller ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_all *Overview:*

Return a list of all the SDN_controllers known to the system.

Signature:

```
SDN_controller ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `SDN_controller ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of SDN_controller references to SDN_controller records for all SDN_controllers known to the system.

Signature:

```
(SDN_controller ref -> SDN_controller record) map get_all_records (
    session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(SDN_controller ref -> SDN_controller record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the SDN_controller instance with the specified UUID.

Signature:

```
SDN_controller ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `SDN_controller ref`

reference to the object

RPC name: get_port *Overview:*

Get the port field of the given SDN_controller.

Signature:

```
int get_port (session ref session_id, SDN_controller ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SDN_controller ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_protocol *Overview:*

Get the protocol field of the given SDN_controller.

Signature:

```
sdn_controller_protocol get_protocol (session ref session_id,
    SDN_controller ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SDN_controller ref`	self	reference to the object

Minimum Role: read-only

Return Type: `sdn_controller_protocol`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given SDN_controller.

Signature:

```
SDN_controller record get_record (session ref session_id,
    SDN_controller ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SDN_controller ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SDN_controller record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given SDN_controller.

Signature:

```
string get_uuid (session ref session_id, SDN_controller ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SDN_controller ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: introduce *Overview:*

Introduce an SDN controller to the pool.

Signature:

```
SDN_controller ref introduce (session ref session_id,
    sdn_controller_protocol protocol, string address, int port)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`sdn_controller_protocol`	protocol	Protocol to connect with the controller.
`string`	address	IP address of the controller.
`int`	port	TCP port of the controller.

Minimum Role: pool-operator

Return Type: `SDN_controller ref`

the introduced SDN controller

Class: secret

A secret

Fields for class: secret

Field	Type	Qualifier	Description
other_config	`(string -> string)map`	*RW*	other_config
uuid	`string`	*RO/runtime*	Unique identifier/object reference
value	`string`	*RW*	the secret

RPCs associated with class: secret

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given secret.

Signature:

```
void add_to_other_config (session ref session_id, secret ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create *Overview:*

Create a new secret instance, and return its handle.

Signature:

```
secret ref create (session ref session_id, secret record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret record`	args	All constructor arguments

Minimum Role: pool-operator

Return Type: `secret ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified secret instance.

Signature:

```
void destroy (session ref session_id, secret ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the secrets known to the system.

Signature:

```
secret ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `secret ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of secret references to secret records for all secrets known to the system.

Signature:

```
(secret ref -> secret record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: pool-operator

Return Type: `(secret ref -> secret record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the secret instance with the specified UUID.

Signature:

```
secret ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: pool-operator

Return Type: `secret ref`

reference to the object

RPC name: get_other_config *Overview:*

Get the other_config field of the given secret.

Signature:

```
(string -> string) map get_other_config (session ref session_id, secret
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given secret.

Signature:

```
secret record get_record (session ref session_id, secret ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: `secret record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given secret.

Signature:

```
string get_uuid (session ref session_id, secret ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: `string`

value of the field

RPC name: get_value *Overview:*

Get the value field of the given secret.

Signature:

```
string get_value (session ref session_id, secret ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given secret. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, secret ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given secret.

Signature:

```
void set_other_config (session ref session_id, secret ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_value *Overview:*

Set the value field of the given secret.

Signature:

```
void set_value (session ref session_id, secret ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`secret ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: session

A session

Fields for class: session

Field	Type	Qualifier	Description
auth_user_name	`string`	*RO/runtime*	the subject name of the user that was externally authenticated. If a session instance has is_local_superuser set, then the value of this field is undefined.
auth_user_sid	`string`	*RO/runtime*	the subject identifier of the user that was externally authenticated. If a session instance has is_local_superuser set, then the value of this field is undefined.
client_certificate	`bool`	*RO/runtime*	indicates whether this session was authenticated using a client certificate
is_local_superuser	`bool`	*RO/runtime*	true iff this session was created using local superuser credentials
last_active	`datetime`	*RO/runtime*	Timestamp for last time session was active
originator	`string`	*RO/runtime*	a key string provided by a API user to distinguish itself from other users sharing the same login name

Field	Type	Qualifier	Description
other_config	`(string -> string)map`	*RW*	additional configuration
parent	`session ref`	*RO/constructor*	references the parent session that created this session
pool	`bool`	*RO/runtime*	True if this session relates to a intra-pool login, false otherwise
rbac_permissions	`string set`	*RO/constructor*	list with all RBAC permissions for this session
subject	`subject ref`	*RO/runtime*	references the subject instance that created the session. If a session instance has is_local_superuser set, then the value of this field is undefined.
tasks	`task ref set`	*RO/runtime*	list of tasks created using the current session
this_host	`host ref`	*RO/runtime*	Currently connected host
this_user	`user ref`	*RO/runtime*	Currently connected user
uuid	`string`	*RO/runtime*	Unique identifier/object reference
validation_time	`datetime`	*RO/runtime*	time when session was last validated

RPCs associated with class: session

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given session.

Signature:

```
void add_to_other_config (session ref session_id, session ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: change_password *Overview:*

Change the account password; if your session is authenticated with root priviledges then the old_pwd is validated and the new_pwd is set regardless

Signature:

```
void change_password (session ref session_id, string old_pwd, string
    new_pwd)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	old_pwd	Old password for account
`string`	new_pwd	New password for account

Return Type: **`void`**

RPC name: create_from_db_file *Overview:*

Signature:

```
session ref create_from_db_file (session ref session_id, string
    filename)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	filename	Database dump filename.

Return Type: `session ref`

ID of newly created session

RPC name: get_all_subject_identifiers *Overview:*

Return a list of all the user subject-identifiers of all existing sessions

Signature:

```
string set get_all_subject_identifiers (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `string set`

The list of user subject-identifiers of all existing sessions

RPC name: get_auth_user_name *Overview:*

Get the auth_user_name field of the given session.

Signature:

```
string get_auth_user_name (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_auth_user_sid *Overview:*

Get the auth_user_sid field of the given session.

Signature:

```
string get_auth_user_sid (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the session instance with the specified UUID.

Signature:

```
session ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `session ref`

reference to the object

RPC name: get_client_certificate *Overview:*

Get the client_certificate field of the given session.

Signature:

```
bool get_client_certificate (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_local_superuser *Overview:*

Get the is_local_superuser field of the given session.

Signature:

```
bool get_is_local_superuser (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_last_active *Overview:*

Get the last_active field of the given session.

Signature:

```
datetime get_last_active (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_originator *Overview:*

Get the originator field of the given session.

Signature:

```
string get_originator (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given session.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_parent *Overview:*

Get the parent field of the given session.

Signature:

```
session ref get_parent (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `session ref`

value of the field

RPC name: get_pool *Overview:*

Get the pool field of the given session.

Signature:

```
bool get_pool (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_rbac_permissions *Overview:*

Get the rbac_permissions field of the given session.

Signature:

```
string set get_rbac_permissions (session ref session_id, session ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given session.

Signature:

```
session record get_record (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `session record`

all fields from the object

RPC name: get_subject *Overview:*

Get the subject field of the given session.

Signature:

```
subject ref get_subject (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `subject ref`

value of the field

RPC name: get_tasks *Overview:*

Get the tasks field of the given session.

Signature:

```
task ref set get_tasks (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `task ref set`

value of the field

RPC name: get_this_host *Overview:*

Get the this_host field of the given session.

Signature:

```
host ref get_this_host (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_this_user *Overview:*

Get the this_user field of the given session.

Signature:

```
user ref get_this_user (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `user ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given session.

Signature:

```
string get_uuid (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_validation_time *Overview:*

Get the validation_time field of the given session.

Signature:

```
datetime get_validation_time (session ref session_id, session ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`session ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: local_logout *Overview:*

Log out of local session.

Signature:

```
void local_logout (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: pool-admin

Return Type: `void`

RPC name: login_with_password *Overview:*

Attempt to authenticate the user, returning a session reference if successful

Signature:

```
session ref login_with_password (string uname, string pwd, string
    version, string originator)
<!--NeedCopy-->
```

Arguments:

type	name	description
`string`	uname	Username for login.
`string`	pwd	Password for login.
`string`	version	Client API version.
`string`	originator	Key string for distinguishing different API users sharing the same login name.

Minimum Role: read-only

Return Type: `session ref`

reference of newly created session

Possible Error Codes: `SESSION_AUTHENTICATION_FAILED`, `HOST_IS_SLAVE`

RPC name: logout *Overview:*

Log out of a session

Signature:

```
void logout (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `void`

RPC name: logout_subject_identifier *Overview:*

Log out all sessions associated to a user subject-identifier, except the session associated with the context calling this function

Signature:

```
void logout_subject_identifier (session ref session_id, string
    subject_identifier)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
string	subject_identifier	User subject-identifier of the sessions to be destroyed

Minimum Role: pool-operator

Return Type: **void**

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given session. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, session ref self
    , string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
session ref	self	reference to the object
string	key	Key to remove

Minimum Role: pool-admin

Return Type: **void**

RPC name: set_other_config *Overview:*

Set the other_config field of the given session.

Signature:

```
void set_other_config (session ref session_id, session ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
session ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

RPC name: slave_local_login_with_password *Overview:*

Authenticate locally against a slave in emergency mode. Note the resulting sessions are only good for use on this host.

Signature:

```
session ref slave_local_login_with_password (string uname, string pwd)
<!--NeedCopy-->
```

Arguments:

type	name	description
string	uname	Username for login.
string	pwd	Password for login.

Minimum Role: pool-admin

Return Type: `session ref`

ID of newly created session

Class: SM

A storage manager plugin

Fields for class: SM

Field	Type	Qualifier	Description
capabilities	`string set`	*RO/runtime*	**Deprecated**. capabilities of the SM plugin
configuration	`(string -> string)map`	*RO/runtime*	names and descriptions of device config keys
copyright	`string`	*RO/runtime*	Entity which owns the copyright of this plugin
driver_filename	`string`	*RO/runtime*	filename of the storage driver
features	`(string -> int) map`	*RO/runtime*	capabilities of the SM plugin, with capability version numbers
name_description	`string`	*RO/runtime*	a notes field containing human-readable description
name_label	`string`	*RO/runtime*	a human-readable name
other_config	`(string -> string)map`	*RW*	additional configuration
required_api_version	`string`	*RO/runtime*	Minimum SM API version required on the server
required_cluster_stack	`string set`	*RO/runtime*	The storage plugin requires that one of these cluster stacks is configured and running.
type	`string`	*RO/runtime*	SR.type
uuid	`string`	*RO/runtime*	Unique identifier/object reference
vendor	`string`	*RO/runtime*	Vendor who created this plugin
version	`string`	*RO/runtime*	Version of the plugin

RPCs associated with class: SM

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given SM.

Signature:

```
void add_to_other_config (session ref session_id, SM ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the SMs known to the system.

Signature:

```
SM ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `SM ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of SM references to SM records for all SMs known to the system.

Signature:

```
(SM ref -> SM record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(SM ref -> SM record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the SM instances with the given label.

Signature:

```
SM ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `SM ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the SM instance with the specified UUID.

Signature:

```
SM ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `SM ref`

reference to the object

RPC name: get_capabilities **This message is deprecated.**

Overview:

Get the capabilities field of the given SM.

Signature:

```
string set get_capabilities (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_configuration *Overview:*

Get the configuration field of the given SM.

Signature:

```
(string -> string) map get_configuration (session ref session_id, SM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_copyright *Overview:*

Get the copyright field of the given SM.

Signature:

```
string get_copyright (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_driver_filename *Overview:*

Get the driver_filename field of the given SM.

Signature:

```
string get_driver_filename (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_features *Overview:*

Get the features field of the given SM.

Signature:

```
(string -> int) map get_features (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> int)map`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given SM.

Signature:

```
string get_name_description (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SM ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given SM.

Signature:

```
string get_name_label (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SM ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given SM.

Signature:

```
(string -> string) map get_other_config (session ref session_id, SM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given SM.

Signature:

```
SM record get_record (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SM record`

all fields from the object

RPC name: get_required_api_version *Overview:*

Get the required_api_version field of the given SM.

Signature:

```
string get_required_api_version (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_required_cluster_stack *Overview:*

Get the required_cluster_stack field of the given SM.

Signature:

```
string set get_required_cluster_stack (session ref session_id, SM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given SM.

Signature:

```
string get_type (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given SM.

Signature:

```
string get_uuid (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor *Overview:*

Get the vendor field of the given SM.

Signature:

```
string get_vendor (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SM ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_version *Overview:*

Get the version field of the given SM.

Signature:

```
string get_version (session ref session_id, SM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SM ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given SM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, SM ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SM ref	self	reference to the object
string	key	Key to remove

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_other_config *Overview:*

Set the other_config field of the given SM.

Signature:

```
void set_other_config (session ref session_id, SM ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SM ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

Class: SR

A storage repository

Fields for class: SR

Field	Type	Qualifier	Description
allowed_operations	`storage_operations set`	*RO/runtime*	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.
blobs	`(string -> blob ref) map`	*RO/runtime*	Binary blobs associated with this SR
clustered	`bool`	*RO/runtime*	True if the SR is using aggregated local storage
content_type	`string`	*RO/constructor*	the type of the SR's content, if required (e.g. ISOs)
current_operations	`(string -> storage_operations) map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
introduced_by	`DR_task ref`	*RO/runtime*	The disaster recovery task which introduced this SR
is_tools_sr	`bool`	*RO/runtime*	True if this is the SR that contains the Tools ISO VDIs
local_cache_enabled	`bool`	*RO/runtime*	True if this SR is assigned to be the local cache for its host
name_description	`string`	*RO/constructor*	a notes field containing human-readable description
name_label	`string`	*RO/constructor*	a human-readable name

Field	Type	Qualifier	Description
other_config	`(string -> string)map`	*RW*	additional configuration
PBDs	`PBD ref set`	*RO/runtime*	describes how particular hosts can see this storage repository
physical_size	`int`	*RO/constructor*	total physical size of the repository (in bytes)
physical_utilisation	`int`	*RO/runtime*	physical space currently utilised on this storage repository (in bytes). Note that for sparse disk formats, physical_utilisation may be less than virtual_allocation
shared	`bool`	*RO/runtime*	true if this SR is (capable of being) shared between multiple hosts
sm_config	`(string -> string)map`	*RW*	SM dependent data
tags	`string set`	*RW*	user-specified tags for categorization purposes
type	`string`	*RO/constructor*	type of the storage repository
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VDIs	`VDI ref set`	*RO/runtime*	all virtual disks known to this storage repository

virtual_allocation	`int`	*RO/runtime*	sum of virtual_sizes of all VDIs in this storage repository (in bytes)

RPCs associated with class: SR

RPC name: add_tags *Overview:*

Add the given value to the tags field of the given SR. If the value is already in that Set, then do nothing.

Signature:

```
void add_tags (session ref session_id, SR ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`string`	value	New value to add

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given SR.

Signature:

```
void add_to_other_config (session ref session_id, SR ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`SR ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: add_to_sm_config *Overview:*

Add the given key-value pair to the sm_config field of the given SR.

Signature:

```
void add_to_sm_config (session ref session_id, SR ref self, string key,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: assert_can_host_ha_statefile *Overview:*

Returns successfully if the given SR can host an HA statefile. Otherwise returns an error to explain why not

Signature:

```
void assert_can_host_ha_statefile (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to query

Minimum Role: pool-operator

Return Type: `void`

RPC name: assert_supports_database_replication *Overview:*

Returns successfully if the given SR supports database replication. Otherwise returns an error to explain why not.

Signature:

```
void assert_supports_database_replication (session ref session_id, SR
    ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to query

Minimum Role: pool-operator

Return Type: `void`

RPC name: create *Overview:*

Create a new Storage Repository and introduce it into the managed system, creating both SR record and PBD record to attach it to current host (with specified device_config parameters)

Signature:

```
SR ref create (session ref session_id, host ref host, (string -> string
    ) map device_config, int physical_size, string name_label, string
    name_description, string type, string content_type, bool shared, (
    string -> string) map sm_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to create/make the SR on
`(string -> string)map`	device_config	The device config string that will be passed to backend SR driver
`int`	physical_size	The physical size of the new storage repository
`string`	name_label	The name of the new storage repository
`string`	name_description	The description of the new storage repository
`string`	type	The type of the SR; used to specify the SR backend driver to use
`string`	content_type	The type of the new SRs content, if required (e.g. ISOs)
`bool`	shared	True if the SR (is capable of) being shared by multiple hosts
`(string -> string)map`	sm_config	Storage backend specific configuration options

Minimum Role: pool-operator

Return Type: `SR ref`

The reference of the newly created Storage Repository.

Possible Error Codes: `SR_UNKNOWN_DRIVER`

RPC name: create_new_blob *Overview:*

Create a placeholder for a named binary blob of data that is associated with this SR

Signature:

```
blob ref create_new_blob (session ref session_id, SR ref sr, string
    name, string mime_type, bool public)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`string`	name	The name associated with the blob
`string`	mime_type	The mime type for the data. Empty string translates to application/octet-stream
`bool`	public	True if the blob should be publicly available

Minimum Role: pool-operator

Return Type: `blob ref`

The reference of the blob, needed for populating its data

RPC name: destroy *Overview:*

Destroy specified SR, removing SR-record from database and remove SR from disk. (In order to affect this operation the appropriate device_config is read from the specified SR's PBD on current host)

Signature:

```
void destroy (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to destroy

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `SR_HAS_PBD`

RPC name: disable_database_replication *Overview:*

Signature:

```
void disable_database_replication (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to which metadata should be no longer replicated

Minimum Role: pool-operator

Return Type: `void`

RPC name: enable_database_replication *Overview:*

Signature:

```
void enable_database_replication (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to which metadata should be replicated

Minimum Role: pool-operator

Return Type: `void`

RPC name: forget *Overview:*

Removing specified SR-record from database, without attempting to remove SR from disk

Signature:

```
void forget (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to destroy

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `SR_HAS_PBD`

RPC name: forget_data_source_archives *Overview:*

Forget the recorded statistics related to the specified data source

Signature:

```
void forget_data_source_archives (session ref session_id, SR ref sr,
    string data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`string`	data_source	The data source whose archives are to be forgotten

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the SRs known to the system.

Signature:

```
SR ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `SR ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of SR references to SR records for all SRs known to the system.

Signature:

```
(SR ref -> SR record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(SR ref -> SR record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given SR.

Signature:

```
storage_operations set get_allowed_operations (session ref session_id,
    SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `storage_operations set`

value of the field

RPC name: get_blobs *Overview:*

Get the blobs field of the given SR.

Signature:

```
(string -> blob ref) map get_blobs (session ref session_id, SR ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> blob ref)map`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the SR instances with the given label.

Signature:

```
SR ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `SR ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the SR instance with the specified UUID.

Signature:

```
SR ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `SR ref`

reference to the object

RPC name: get_clustered *Overview:*

Get the clustered field of the given SR.

Signature:

```
bool get_clustered (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_content_type *Overview:*

Get the content_type field of the given SR.

Signature:

```
string get_content_type (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given SR.

Signature:

```
(string -> storage_operations) map get_current_operations (session ref
   session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> storage_operations)map`

value of the field

RPC name: get_data_sources *Overview:*

Signature:

```
data_source record set get_data_sources (session ref session_id, SR ref
    sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to interrogate

Minimum Role: read-only

Return Type: `data_source record set`

A set of data sources

RPC name: get_introduced_by *Overview:*

Get the introduced_by field of the given SR.

Signature:

```
DR_task ref get_introduced_by (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `DR_task ref`

value of the field

RPC name: get_is_tools_sr *Overview:*

Get the is_tools_sr field of the given SR.

Signature:

```
bool get_is_tools_sr (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_local_cache_enabled *Overview:*

Get the local_cache_enabled field of the given SR.

Signature:

```
bool get_local_cache_enabled (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given SR.

Signature:

```
string get_name_description (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given SR.

Signature:

```
string get_name_label (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given SR.

Signature:

```
(string -> string) map get_other_config (session ref session_id, SR ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PBDs *Overview:*

Get the PBDs field of the given SR.

Signature:

```
PBD ref set get_PBDs (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PBD ref set`

value of the field

RPC name: get_physical_size *Overview:*

Get the physical_size field of the given SR.

Signature:

```
int get_physical_size (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_physical_utilisation *Overview:*

Get the physical_utilisation field of the given SR.

Signature:

```
int get_physical_utilisation (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given SR.

Signature:

```
SR record get_record (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR record`

all fields from the object

RPC name: get_shared *Overview:*

Get the shared field of the given SR.

Signature:

```
bool get_shared (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_sm_config *Overview:*

Get the sm_config field of the given SR.

Signature:

```
(string -> string) map get_sm_config (session ref session_id, SR ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_supported_types *Overview:*

Return a set of all the SR types supported by the system

Signature:

```
string set get_supported_types (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `string set`

the supported SR types

RPC name: get_tags *Overview:*

Get the tags field of the given SR.

Signature:

```
string set get_tags (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given SR.

Signature:

```
string get_type (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given SR.

Signature:

```
string get_uuid (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VDIs *Overview:*

Get the VDIs field of the given SR.

Signature:

```
VDI ref set get_VDIs (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref set`

value of the field

RPC name: get_virtual_allocation *Overview:*

Get the virtual_allocation field of the given SR.

Signature:

```
int get_virtual_allocation (session ref session_id, SR ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`int`**

value of the field

RPC name: introduce *Overview:*

Introduce a new Storage Repository into the managed system

Signature:

```
SR ref introduce (session ref session_id, string uuid, string
    name_label, string name_description, string type, string
    content_type, bool shared, (string -> string) map sm_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	The uuid assigned to the introduced SR
`string`	name_label	The name of the new storage repository
`string`	name_description	The description of the new storage repository
`string`	type	The type of the SR; used to specify the SR backend driver to use
`string`	content_type	The type of the new SRs content, if required (e.g. ISOs)
`bool`	shared	True if the SR (is capable of) being shared by multiple hosts

type	name	description
(string -> string)map	sm_config	Storage backend specific configuration options

Minimum Role: pool-operator

Return Type: SR ref

The reference of the newly introduced Storage Repository.

RPC name: make This message is deprecated.

Overview:

Create a new Storage Repository on disk. This call is deprecated: use SR.create instead.

Signature:

```
string make (session ref session_id, host ref host, (string -> string)
    map device_config, int physical_size, string name_label, string
    name_description, string type, string content_type, (string ->
    string) map sm_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host ref	host	The host to create/make the SR on
(string -> string)map	device_config	The device config string that will be passed to backend SR driver
int	physical_size	The physical size of the new storage repository
string	name_label	The name of the new storage repository
string	name_description	The description of the new storage repository
string	type	The type of the SR; used to specify the SR backend driver to use

type	name	description
`string`	content_type	The type of the new SRs content, if required (e.g. ISOs)
`(string -> string)map`	sm_config	Storage backend specific configuration options

Minimum Role: pool-operator

Return Type: `string`

The uuid of the newly created Storage Repository.

RPC name: probe *Overview:*

Perform a backend-specific scan, using the given device_config. If the device_config is complete, then this will return a list of the SRs present of this type on the device, if any. If the device_config is partial, then a backend-specific scan will be performed, returning results that will guide the user in improving the device_config.

Signature:

```
string probe (session ref session_id, host ref host, (string -> string)
    map device_config, string type, (string -> string) map sm_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to create/make the SR on
`(string -> string)map`	device_config	The device config string that will be passed to backend SR driver
`string`	type	The type of the SR; used to specify the SR backend driver to use
`(string -> string)map`	sm_config	Storage backend specific configuration options

Minimum Role: pool-operator

Return Type: `string`

An XML fragment containing the scan results. These are specific to the scan being performed, and the backend.

RPC name: probe_ext *Overview:*

Perform a backend-specific scan, using the given device_config. If the device_config is complete, then this will return a list of the SRs present of this type on the device, if any. If the device_config is partial, then a backend-specific scan will be performed, returning results that will guide the user in improving the device_config.

Signature:

```
probe_result record set probe_ext (session ref session_id, host ref
    host, (string -> string) map device_config, string type, (string ->
    string) map sm_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`host ref`	host	The host to create/make the SR on
`(string -> string)map`	device_config	The device config string that will be passed to backend SR driver
`string`	type	The type of the SR; used to specify the SR backend driver to use
`(string -> string)map`	sm_config	Storage backend specific configuration options

Minimum Role: pool-operator

Return Type: `probe_result record set`

A set of records containing the scan results.

RPC name: query_data_source *Overview:*

Query the latest value of the specified data source

Signature:

```
float query_data_source (session ref session_id, SR ref sr, string
    data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`string`	data_source	The data source to query

Minimum Role: read-only

Return Type: **`float`**

The latest value, averaged over the last 5 seconds

RPC name: record_data_source *Overview:*

Start recording the specified data source

Signature:

```
void record_data_source (session ref session_id, SR ref sr, string
    data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`string`	data_source	The data source to record

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given SR. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, SR ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_from_sm_config *Overview:*

Remove the given key and its corresponding value from the sm_config field of the given SR. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_sm_config (session ref session_id, SR ref self, string
    key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_tags *Overview:*

Remove the given value from the tags field of the given SR. If the value is not in that Set, then do nothing.

Signature:

```
void remove_tags (session ref session_id, SR ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`string`	value	Value to remove

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: scan *Overview:*

Refreshes the list of VDIs associated with an SR

Signature:

```
void scan (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR to scan

Minimum Role: vm-power-admin

Return Type: **`void`**

RPC name: set_name_description *Overview:*

Set the name description of the SR

Signature:

```
void set_name_description (session ref session_id, SR ref sr, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`string`	value	The name description for the SR

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_name_label *Overview:*

Set the name label of the SR

Signature:

```
void set_name_label (session ref session_id, SR ref sr, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`string`	value	The name label for the SR

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given SR.

Signature:

```
void set_other_config (session ref session_id, SR ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SR ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_physical_size *Overview:*

Sets the SR's physical_size field

Signature:

```
void set_physical_size (session ref session_id, SR ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
SR ref	self	The SR to modify
int	value	The new value of the SR's physical_size

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_shared *Overview:*

Sets the shared flag on the SR

Signature:

```
void set_shared (session ref session_id, SR ref sr, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR
`bool`	value	True if the SR is shared

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_sm_config *Overview:*

Set the sm_config field of the given SR.

Signature:

```
void set_sm_config (session ref session_id, SR ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_tags *Overview:*

Set the tags field of the given SR.

Signature:

```
void set_tags (session ref session_id, SR ref self, string set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	self	reference to the object
`string set`	value	New value to set

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: update *Overview:*

Refresh the fields on the SR object

Signature:

```
void update (session ref session_id, SR ref sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`SR ref`	sr	The SR whose fields should be refreshed

Minimum Role: pool-operator

Return Type: **`void`**

Class: sr_stat

A set of high-level properties associated with an SR.

Fields for class: sr_stat

Field	Type	Qualifier	Description
clustered	`bool`	*RO/runtime*	Indicates whether the SR uses clustered local storage.
free_space	`int`	*RO/runtime*	Number of bytes free on the backing storage (in bytes)
health	`sr_health`	*RO/runtime*	The health status of the SR.
name_description	`string`	*RO/runtime*	Longer, human-readable description of the SR. Descriptions are generally only displayed by clients when the user is examining SRs in detail.
name_label	`string`	*RO/runtime*	Short, human-readable label for the SR.
total_space	`int`	*RO/runtime*	Total physical size of the backing storage (in bytes)
uuid	`string option`	*RO/runtime*	Uuid that uniquely identifies this SR, if one is available.

RPCs associated with class: sr_stat

Class sr_stat has no additional RPCs associated with it.

Class: subject

A user or group that can log in xapi

Fields for class: subject

Field	Type	Qualifier	Description
other_config	`(string -> string)map`	*RO/constructor*	additional configuration
roles	`role ref set`	*RO/runtime*	the roles associated with this subject
subject_identifier	`string`	*RO/constructor*	the subject identifier, unique in the external directory service
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: subject

RPC name: add_to_roles *Overview:*

This call adds a new role to a subject

Signature:

```
void add_to_roles (session ref session_id, subject ref self, role ref
    role)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	The subject who we want to add the role to
`role ref`	role	The unique role reference

Minimum Role: pool-admin

Return Type: `void`

RPC name: create *Overview:*

Create a new subject instance, and return its handle.

Signature:

```
subject ref create (session ref session_id, subject record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject record`	args	All constructor arguments

Minimum Role: pool-admin

Return Type: `subject ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified subject instance.

Signature:

```
void destroy (session ref session_id, subject ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	reference to the object

Minimum Role: pool-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the subjects known to the system.

Signature:

```
subject ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `subject ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of subject references to subject records for all subjects known to the system.

Signature:

```
(subject ref -> subject record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(subject ref -> subject record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the subject instance with the specified UUID.

Signature:

```
subject ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `subject ref`

reference to the object

RPC name: get_other_config *Overview:*

Get the other_config field of the given subject.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    subject ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_permissions_name_label *Overview:*

This call returns a list of permission names given a subject

Signature:

```
string set get_permissions_name_label (session ref session_id, subject
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	The subject whose permissions will be retrieved

Minimum Role: read-only

Return Type: `string set`

a list of permission names

RPC name: get_record *Overview:*

Get a record containing the current state of the given subject.

Signature:

```
subject record get_record (session ref session_id, subject ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	reference to the object

Minimum Role: read-only

Return Type: `subject record`

all fields from the object

RPC name: get_roles *Overview:*

Get the roles field of the given subject.

Signature:

```
role ref set get_roles (session ref session_id, subject ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	reference to the object

Minimum Role: read-only

Return Type: `role ref set`

value of the field

RPC name: get_subject_identifier *Overview:*

Get the subject_identifier field of the given subject.

Signature:

```
string get_subject_identifier (session ref session_id, subject ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given subject.

Signature:

```
string get_uuid (session ref session_id, subject ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_roles *Overview:*

This call removes a role from a subject

Signature:

```
void remove_from_roles (session ref session_id, subject ref self, role
    ref role)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`subject ref`	self	The subject from whom we want to remove the role
`role ref`	role	The unique role reference in the subject's roles field

Minimum Role: pool-admin

Return Type: **`void`**

Class: task

A long-running asynchronous task

Fields for class: task

Field	Type	Qualifier	Description
allowed_operations	`task_allowed_operations set`	*RO/runtime*	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.
backtrace	`string`	*RO/runtime*	Function call trace for debugging.

Field	Type	Qualifier	Description
created	`datetime`	*RO/runtime*	Time task was created
current_operations	`(string -> task_allowed_operations )map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
error_info	`string set`	*RO/runtime*	if the task has failed, this field contains the set of associated error strings. Undefined otherwise.
finished	`datetime`	*RO/runtime*	Time task finished (i.e. succeeded or failed). If task-status is pending, then the value of this field has no meaning
name_description	`string`	*RO/runtime*	a notes field containing human-readable description
name_label	`string`	*RO/runtime*	a human-readable name
other_config	`(string -> string)map`	*RW*	additional configuration
progress	**`float`**	*RO/runtime*	This field contains the estimated fraction of the task which is complete. This field should not be used to determine whether the task is complete - for this the status field of the task should be used.

Field	Type	Qualifier	Description
resident_on	`host ref`	*RO/runtime*	the host on which the task is running
result	`string`	*RO/runtime*	if the task has completed successfully, this field contains the result value (either Void or an object reference). Undefined otherwise.
status	`task_status_type`	*RO/runtime*	current status of the task
subtask_of	`task ref`	*RO/runtime*	Ref pointing to the task this is a substask of.
subtasks	`task ref set`	*RO/runtime*	List pointing to all the substasks.
type	`string`	*RO/runtime*	if the task has completed successfully, this field contains the type of the encoded result (i.e. name of the class whose reference is in the result field). Undefined otherwise.
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: task

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given task.

Signature:

```
void add_to_other_config (session ref session_id, task ref self, string
     key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: cancel *Overview:*

Request that a task be cancelled. Note that a task may fail to be cancelled and may complete or fail normally and note that, even when a task does cancel, it might take an arbitrary amount of time.

Signature:

```
void cancel (session ref session_id, task ref task)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	task	The task

Minimum Role: read-only

Return Type: **`void`**

Possible Error Codes: `OPERATION_NOT_ALLOWED`

RPC name: create *Overview:*

Create a new task object which must be manually destroyed.

Signature:

```
task ref create (session ref session_id, string label, string
    description)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	short label for the new task
`string`	description	longer description for the new task

Minimum Role: read-only

Return Type: `task ref`

The reference of the created task object

RPC name: destroy *Overview:*

Destroy the task object

Signature:

```
void destroy (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	Reference to the task object

Minimum Role: read-only

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the tasks known to the system.

Signature:

```
task ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `task ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of task references to task records for all tasks known to the system.

Signature:

```
(task ref -> task record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(task ref -> task record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given task.

Signature:

```
task_allowed_operations set get_allowed_operations (session ref
    session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `task_allowed_operations set`

value of the field

RPC name: get_backtrace *Overview:*

Get the backtrace field of the given task.

Signature:

```
string get_backtrace (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the task instances with the given label.

Signature:

```
task ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `task ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the task instance with the specified UUID.

Signature:

```
task ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `task ref`

reference to the object

RPC name: get_created *Overview:*

Get the created field of the given task.

Signature:

```
datetime get_created (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given task.

Signature:

```
(string -> task_allowed_operations) map get_current_operations (session
    ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> task_allowed_operations)map`

value of the field

RPC name: get_error_info *Overview:*

Get the error_info field of the given task.

Signature:

```
string set get_error_info (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_finished *Overview:*

Get the finished field of the given task.

Signature:

```
datetime get_finished (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given task.

Signature:

```
string get_name_description (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given task.

Signature:

```
string get_name_label (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given task.

Signature:

```
(string -> string) map get_other_config (session ref session_id, task
   ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_progress *Overview:*

Get the progress field of the given task.

Signature:

```
float get_progress (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `float`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given task.

Signature:

```
task record get_record (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `task record`

all fields from the object

RPC name: get_resident_on *Overview:*

Get the resident_on field of the given task.

Signature:

```
host ref get_resident_on (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_result *Overview:*

Get the result field of the given task.

Signature:

```
string get_result (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_status *Overview:*

Get the status field of the given task.

Signature:

```
task_status_type get_status (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `task_status_type`

value of the field

RPC name: get_subtask_of *Overview:*

Get the subtask_of field of the given task.

Signature:

```
task ref get_subtask_of (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `task ref`

value of the field

RPC name: get_subtasks *Overview:*

Get the subtasks field of the given task.

Signature:

```
task ref set get_subtasks (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `task ref set`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given task.

Signature:

```
string get_type (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given task.

Signature:

```
string get_uuid (session ref session_id, task ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given task. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, task ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_error_info *Overview:*

Set the task error info

Signature:

```
void set_error_info (session ref session_id, task ref self, string set
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	Reference to the task object
`string set`	value	Task error info to be set

Minimum Role: read-only

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given task.

Signature:

```
void set_other_config (session ref session_id, task ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_progress *Overview:*

Set the task progress

Signature:

```
void set_progress (session ref session_id, task ref self, float value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	Reference to the task object
`float`	value	Task progress value to be set

Minimum Role: read-only

Return Type: `void`

RPC name: set_result *Overview:*

Set the task result

Signature:

```
void set_result (session ref session_id, task ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	Reference to the task object
`string`	value	Task result to be set

Minimum Role: read-only

Return Type: `void`

RPC name: set_status *Overview:*

Set the task status

Signature:

```
void set_status (session ref session_id, task ref self,
    task_status_type value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`task ref`	self	Reference to the task object
`task_status_type`	value	task status value to be set

Minimum Role: read-only

Return Type: `void`

Class: tunnel

A tunnel for network traffic

Fields for class: tunnel

Field	Type	Qualifier	Description
access_PIF	`PIF ref`	*RO/constructor*	The interface through which the tunnel is accessed
other_config	`(string -> string)map`	*RW*	Additional configuration
protocol	`tunnel_protocol`	*RW*	The protocol used for tunneling (either GRE or VxLAN)
status	`(string -> string)map`	*RW*	Status information about the tunnel

Field	Type	Qualifier	Description
transport_PIF	PIF ref	*RO/constructor*	The interface used by the tunnel
uuid	string	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: tunnel

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given tunnel.

Signature:

```
void add_to_other_config (session ref session_id, tunnel ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
tunnel ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: pool-operator

Return Type: `void`

RPC name: add_to_status *Overview:*

Add the given key-value pair to the status field of the given tunnel.

Signature:

```
void add_to_status (session ref session_id, tunnel ref self, string key
    , string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create *Overview:*

Create a tunnel

Signature:

```
tunnel ref create (session ref session_id, PIF ref transport_PIF,
    network ref network, tunnel_protocol protocol)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	transport_PIF	PIF which receives the tagged traffic
`network ref`	network	Network to receive the tunnelled traffic
`tunnel_protocol`	protocol	Protocol used for the tunnel (GRE or VxLAN)

Minimum Role: pool-operator

Return Type: `tunnel ref`

The reference of the created tunnel object

Possible Error Codes: `OPENVSWITCH_NOT_ACTIVE`, `TRANSPORT_PIF_NOT_CONFIGURED`, `IS_TUNNEL_ACCESS_PIF`

RPC name: destroy *Overview:*

Destroy a tunnel

Signature:

```
void destroy (session ref session_id, tunnel ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	tunnel to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_access_PIF *Overview:*

Get the access_PIF field of the given tunnel.

Signature:

```
PIF ref get_access_PIF (session ref session_id, tunnel ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_all *Overview:*

Return a list of all the tunnels known to the system.

Signature:

```
tunnel ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `tunnel ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of tunnel references to tunnel records for all tunnels known to the system.

Signature:

```
(tunnel ref -> tunnel record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(tunnel ref -> tunnel record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the tunnel instance with the specified UUID.

Signature:

```
tunnel ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `tunnel ref`

reference to the object

RPC name: get_other_config *Overview:*

Get the other_config field of the given tunnel.

Signature:

```
(string -> string) map get_other_config (session ref session_id, tunnel
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_protocol *Overview:*

Get the protocol field of the given tunnel.

Signature:

```
tunnel_protocol get_protocol (session ref session_id, tunnel ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `tunnel_protocol`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given tunnel.

Signature:

```
tunnel record get_record (session ref session_id, tunnel ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `tunnel record`

all fields from the object

RPC name: get_status *Overview:*

Get the status field of the given tunnel.

Signature:

```
(string -> string) map get_status (session ref session_id, tunnel ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_transport_PIF *Overview:*

Get the transport_PIF field of the given tunnel.

Signature:

```
PIF ref get_transport_PIF (session ref session_id, tunnel ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given tunnel.

Signature:

```
string get_uuid (session ref session_id, tunnel ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given tunnel. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, tunnel ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_from_status *Overview:*

Remove the given key and its corresponding value from the status field of the given tunnel. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_status (session ref session_id, tunnel ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given tunnel.

Signature:

```
void set_other_config (session ref session_id, tunnel ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_protocol *Overview:*

Set the protocol field of the given tunnel.

Signature:

```
void set_protocol (session ref session_id, tunnel ref self,
    tunnel_protocol value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object
`tunnel_protocol`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_status *Overview:*

Set the status field of the given tunnel.

Signature:

```
void set_status (session ref session_id, tunnel ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`tunnel ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

Class: USB_group

A group of compatible USBs across the resource pool

Fields for class: USB_group

Field	Type	Qualifier	Description
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
other_config	`(string -> string)map`	*RW*	Additional configuration
PUSBs	`PUSB ref set`	*RO/runtime*	List of PUSBs in the group

Field	Type	Qualifier	Description
uuid	string	*RO/runtime*	Unique identifier/object reference
VUSBs	VUSB ref set	*RO/runtime*	List of VUSBs using the group

RPCs associated with class: USB_group

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given USB_group.

Signature:

```
void add_to_other_config (session ref session_id, USB_group ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
USB_group ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: pool-admin

Return Type: **void**

RPC name: create *Overview:*

Signature:

```
USB_group ref create (session ref session_id, string name_label, string
     name_description, (string -> string) map other_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
string	name_label	
string	name_description	
(string -> string)map	other_config	

Minimum Role: pool-admin

Return Type: `USB_group ref`

RPC name: destroy *Overview:*

Signature:

```
void destroy (session ref session_id, USB_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
USB_group ref	self	The USB group to destroy

Minimum Role: pool-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the USB_groups known to the system.

Signature:

```
USB_group ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `USB_group ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of USB_group references to USB_group records for all USB_groups known to the system.

Signature:

```
(USB_group ref -> USB_group record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(USB_group ref -> USB_group record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the USB_group instances with the given label.

Signature:

```
USB_group ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `USB_group ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the USB_group instance with the specified UUID.

Signature:

```
USB_group ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
string	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `USB_group ref`

reference to the object

RPC name: get_name_description *Overview:*

Get the name/description field of the given USB_group.

Signature:

```
string get_name_description (session ref session_id, USB_group ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
USB_group ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given USB_group.

Signature:

```
string get_name_label (session ref session_id, USB_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given USB_group.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    USB_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PUSBs *Overview:*

Get the PUSBs field of the given USB_group.

Signature:

```
PUSB ref set get_PUSBs (session ref session_id, USB_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PUSB ref set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given USB_group.

Signature:

```
USB_group record get_record (session ref session_id, USB_group ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `USB_group record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given USB_group.

Signature:

```
string get_uuid (session ref session_id, USB_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VUSBs *Overview:*

Get the VUSBs field of the given USB_group.

Signature:

```
VUSB ref set get_VUSBs (session ref session_id, USB_group ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VUSB ref set`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given USB_group. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, USB_group ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_name_description *Overview:*

Set the name/description field of the given USB_group.

Signature:

```
void set_name_description (session ref session_id, USB_group ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name/label field of the given USB_group.

Signature:

```
void set_name_label (session ref session_id, USB_group ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given USB_group.

Signature:

```
void set_other_config (session ref session_id, USB_group ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`USB_group ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-admin

Return Type: **`void`**

Class: user

This class is deprecated.

A user of the system

Fields for class: user

Field	Type	Qualifier	Description
fullname	`string`	*RW*	**Deprecated**. full name
other_config	`(string -> string)map`	*RW*	**Deprecated**. additional configuration
short_name	`string`	*RO/constructor*	**Deprecated**. short name (e.g. userid)
uuid	`string`	*RO/runtime*	**Deprecated**. Unique identifier/object reference

RPCs associated with class: user

RPC name: add_to_other_config   This message is deprecated.

Overview:

Add the given key-value pair to the other_config field of the given user.

Signature:

```
void add_to_other_config (session ref session_id, user ref self, string
     key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: create **This message is deprecated.**

Overview:

Create a new user instance, and return its handle.

Signature:

```
user ref create (session ref session_id, user record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user record`	args	All constructor arguments

Minimum Role: pool-admin

Return Type: `user ref`

reference to the newly created object

RPC name: destroy **This message is deprecated.**

Overview:

Destroy the specified user instance.

Signature:

```
void destroy (session ref session_id, user ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object

Minimum Role: pool-admin

Return Type: `void`

RPC name: get_by_uuid   This message is deprecated.

Overview:

Get a reference to the user instance with the specified UUID.

Signature:

```
user ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `user ref`

reference to the object

RPC name: get_fullname   This message is deprecated.

Overview:

Get the fullname field of the given user.

Signature:

```
string get_fullname (session ref session_id, user ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_other_config **This message is deprecated.**

Overview:

Get the other_config field of the given user.

Signature:

```
(string -> string) map get_other_config (session ref session_id, user
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record **This message is deprecated.**

Overview:

Get a record containing the current state of the given user.

Signature:

```
user record get_record (session ref session_id, user ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object

Minimum Role: read-only

Return Type: `user record`

all fields from the object

RPC name: get_short_name   This message is deprecated.

Overview:

Get the short_name field of the given user.

Signature:

```
string get_short_name (session ref session_id, user ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid   This message is deprecated.

Overview:

Get the uuid field of the given user.

Signature:

```
string get_uuid (session ref session_id, user ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config This message is deprecated.

Overview:

Remove the given key and its corresponding value from the other_config field of the given user. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, user ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_fullname This message is deprecated.

Overview:

Set the fullname field of the given user.

Signature:

```
void set_fullname (session ref session_id, user ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

RPC name: set_other_config **This message is deprecated.**

Overview:

Set the other_config field of the given user.

Signature:

```
void set_other_config (session ref session_id, user ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`user ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

Class: VBD

A virtual block device

Fields for class: VBD

Field	Type	Qualifier	Description
allowed_operations	`vbd_operations set`	*RO/runtime*	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.
bootable	`bool`	*RW*	true if this VBD is bootable

Field	Type	Qualifier	Description
current_operations	(string -> vbd_operations) map	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
currently_attached	bool	*RO/constructor*	is the device currently attached (erased on reboot)
device	string	*RO/constructor*	device seen by the guest e.g. hda1
empty	bool	*RO/constructor*	if true this represents an empty drive
metrics	VBD_metrics ref	*RO/runtime*	**Removed**. metrics associated with this VBD
mode	vbd_mode	*RO/constructor*	the mode the VBD should be mounted with
other_config	(string -> string) map	*RW*	additional configuration
qos_algorithm_params	(string -> string) map	*RW*	parameters for chosen QoS algorithm
qos_algorithm_type	string	*RW*	QoS algorithm to use
qos_supported_algorithms	string set	*RO/runtime*	supported QoS algorithms for this VBD
runtime_properties	(string -> string) map	*RO/runtime*	Device runtime properties
status_code	int	*RO/runtime*	error/success code associated with last attach-operation (erased on reboot)

Field	Type	Qualifier	Description
status_detail	string	*RO/runtime*	error/success information associated with last attach-operation status (erased on reboot)
storage_lock	bool	*RO/runtime*	true if a storage level lock was acquired
type	vbd_type	*RW*	how the VBD will appear to the guest (e.g. disk or CD)
unpluggable	bool	*RW*	true if this VBD will support hot-unplug
userdevice	string	*RW*	user-friendly device name e.g. 0,1,2,etc.
uuid	string	*RO/runtime*	Unique identifier/object reference
VDI	VDI ref	*RO/constructor*	the virtual disk
VM	VM ref	*RO/constructor*	the virtual machine

RPCs associated with class: VBD

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VBD.

Signature:

```
void add_to_other_config (session ref session_id, VBD ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VBD ref	self	reference to the object

type	name	description
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_qos_algorithm_params *Overview:*

Add the given key-value pair to the qos/algorithm_params field of the given VBD.

Signature:

```
void add_to_qos_algorithm_params (session ref session_id, VBD ref self,
     string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: assert_attachable *Overview:*

Throws an error if this VBD could not be attached to this VM if the VM were running. Intended for debugging.

Signature:

```
void assert_attachable (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	The VBD to query

Minimum Role: vm-admin

Return Type: `void`

RPC name: create *Overview:*

Create a new VBD instance, and return its handle.

Signature:

```
VBD ref create (session ref session_id, VBD record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD record`	args	All constructor arguments

Minimum Role: vm-admin

Return Type: `VBD ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified VBD instance.

Signature:

```
void destroy (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: eject *Overview:*

Remove the media from the device and leave it empty

Signature:

```
void eject (session ref session_id, VBD ref vbd)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	vbd	The vbd representing the CDROM-like device

Minimum Role: vm-operator

Return Type: **`void`**

Possible Error Codes: `VBD_NOT_REMOVABLE_MEDIA`, `VBD_IS_EMPTY`

RPC name: get_all *Overview:*

Return a list of all the VBDs known to the system.

Signature:

```
VBD ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VBD ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VBD references to VBD records for all VBDs known to the system.

Signature:

```
(VBD ref -> VBD record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VBD ref -> VBD record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VBD.

Signature:

```
vbd_operations set get_allowed_operations (session ref session_id, VBD
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vbd_operations set`

value of the field

RPC name: get_bootable *Overview:*

Get the bootable field of the given VBD.

Signature:

```
bool get_bootable (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the VBD instance with the specified UUID.

Signature:

```
VBD ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VBD ref`

reference to the object

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VBD.

Signature:

```
(string -> vbd_operations) map get_current_operations (session ref
    session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vbd_operations)map`

value of the field

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given VBD.

Signature:

```
bool get_currently_attached (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_device *Overview:*

Get the device field of the given VBD.

Signature:

```
string get_device (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_empty *Overview:*

Get the empty field of the given VBD.

Signature:

```
bool get_empty (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_metrics **This message is removed.**

Overview:

Get the metrics field of the given VBD.

Signature:

```
VBD_metrics ref get_metrics (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VBD_metrics ref`

value of the field

RPC name: get_mode *Overview:*

Get the mode field of the given VBD.

Signature:

```
vbd_mode get_mode (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vbd_mode`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VBD.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VBD
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_qos_algorithm_params *Overview:*

Get the qos/algorithm_params field of the given VBD.

Signature:

```
(string -> string) map get_qos_algorithm_params (session ref session_id
    , VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_qos_algorithm_type *Overview:*

Get the qos/algorithm_type field of the given VBD.

Signature:

```
string get_qos_algorithm_type (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VBD ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_qos_supported_algorithms *Overview:*

Get the qos/supported_algorithms field of the given VBD.

Signature:

```
string set get_qos_supported_algorithms (session ref session_id, VBD
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VBD ref	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VBD.

Signature:

```
VBD record get_record (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VBD record`

all fields from the object

RPC name: get_runtime_properties *Overview:*

Get the runtime_properties field of the given VBD.

Signature:

```
(string -> string) map get_runtime_properties (session ref session_id,
    VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_status_code *Overview:*

Get the status_code field of the given VBD.

Signature:

```
int get_status_code (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`int`**

value of the field

RPC name: get_status_detail *Overview:*

Get the status_detail field of the given VBD.

Signature:

```
string get_status_detail (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_storage_lock *Overview:*

Get the storage_lock field of the given VBD.

Signature:

```
bool get_storage_lock (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given VBD.

Signature:

```
vbd_type get_type (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vbd_type`

value of the field

RPC name: get_unpluggable *Overview:*

Get the unpluggable field of the given VBD.

Signature:

```
bool get_unpluggable (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_userdevice *Overview:*

Get the userdevice field of the given VBD.

Signature:

```
string get_userdevice (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VBD.

Signature:

```
string get_uuid (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VDI *Overview:*

Get the VDI field of the given VBD.

Signature:

```
VDI ref get_VDI (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

RPC name: get_VM *Overview:*

Get the VM field of the given VBD.

Signature:

```
VM ref get_VM (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: insert *Overview:*

Insert new media into the device

Signature:

```
void insert (session ref session_id, VBD ref vbd, VDI ref vdi)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	vbd	The vbd representing the CDROM-like device
`VDI ref`	vdi	The new VDI to 'insert'

Minimum Role: vm-operator

Return Type: **`void`**

Possible Error Codes: `VBD_NOT_REMOVABLE_MEDIA`, `VBD_NOT_EMPTY`

RPC name: plug *Overview:*

Hotplug the specified VBD, dynamically attaching it to the running VM

Signature:

```
void plug (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	The VBD to hotplug

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VBD. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VBD ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_qos_algorithm_params *Overview:*

Remove the given key and its corresponding value from the qos/algorithm_params field of the given VBD. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_qos_algorithm_params (session ref session_id, VBD ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_bootable *Overview:*

Set the bootable field of the given VBD.

Signature:

```
void set_bootable (session ref session_id, VBD ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_mode *Overview:*

Sets the mode of the VBD. The power_state of the VM must be halted.

Signature:

```
void set_mode (session ref session_id, VBD ref self, vbd_mode value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	Reference to the object
`vbd_mode`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VBD.

Signature:

```
void set_other_config (session ref session_id, VBD ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_qos_algorithm_params *Overview:*

Set the qos/algorithm_params field of the given VBD.

Signature:

```
void set_qos_algorithm_params (session ref session_id, VBD ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_qos_algorithm_type *Overview:*

Set the qos/algorithm_type field of the given VBD.

Signature:

```
void set_qos_algorithm_type (session ref session_id, VBD ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_type *Overview:*

Set the type field of the given VBD.

Signature:

```
void set_type (session ref session_id, VBD ref self, vbd_type value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`vbd_type`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_unpluggable *Overview:*

Set the unpluggable field of the given VBD.

Signature:

```
void set_unpluggable (session ref session_id, VBD ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_userdevice *Overview:*

Set the userdevice field of the given VBD.

Signature:

```
void set_userdevice (session ref session_id, VBD ref self, string value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: unplug *Overview:*

Hot-unplug the specified VBD, dynamically unattaching it from the running VM

Signature:

```
void unplug (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	The VBD to hot-unplug

Minimum Role: vm-admin

Return Type: **`void`**

Possible Error Codes: `DEVICE_DETACH_REJECTED`, `DEVICE_ALREADY_DETACHED`

RPC name: unplug_force *Overview:*

Forcibly unplug the specified VBD

Signature:

```
void unplug_force (session ref session_id, VBD ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD ref`	self	The VBD to forcibly unplug

Minimum Role: vm-admin

Return Type: **`void`**

Class: VBD_metrics

This class is removed.

The metrics associated with a virtual block device

Fields for class: VBD_metrics

Field	Type	Qualifier	Description
io_read_kbs	**`float`**	*RO/runtime*	**Removed**. Read bandwidth (KiB/s)
io_write_kbs	**`float`**	*RO/runtime*	**Removed**. Write bandwidth (KiB/s)
last_updated	`datetime`	*RO/runtime*	**Removed**. Time at which this information was last updated
other_config	`(string -> string)map`	*RW*	**Removed**. additional configuration
uuid	`string`	*RO/runtime*	**Removed**. Unique identifier/object reference

RPCs associated with class: VBD_metrics

RPC name: add_to_other_config This message is removed.

Overview:

Add the given key-value pair to the other_config field of the given VBD_metrics.

Signature:

```
void add_to_other_config (session ref session_id, VBD_metrics ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: get_all This message is removed.

Overview:

Return a list of all the VBD_metrics instances known to the system.

Signature:

```
VBD_metrics ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VBD_metrics ref set`

references to all objects

RPC name: get_all_records This message is removed.

Overview:

Return a map of VBD_metrics references to VBD_metrics records for all VBD_metrics instances known to the system.

Signature:

```
(VBD_metrics ref -> VBD_metrics record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VBD_metrics ref -> VBD_metrics record)map`

records of all objects

RPC name: get_by_uuid This message is removed.

Overview:

Get a reference to the VBD_metrics instance with the specified UUID.

Signature:

```
VBD_metrics ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VBD_metrics ref`

reference to the object

RPC name: get_io_read_kbs This message is removed.

Overview:

Get the io/read_kbs field of the given VBD_metrics.

Signature:

```
float get_io_read_kbs (session ref session_id, VBD_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`float`**

value of the field

RPC name: get_io_write_kbs This message is removed.

Overview:

Get the io/write_kbs field of the given VBD_metrics.

Signature:

```
float get_io_write_kbs (session ref session_id, VBD_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`float`**

value of the field

RPC name: get_last_updated This message is removed.

Overview:

Get the last_updated field of the given VBD_metrics.

Signature:

```
datetime get_last_updated (session ref session_id, VBD_metrics ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_other_config   This message is removed.

Overview:

Get the other_config field of the given VBD_metrics.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    VBD_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record   This message is removed.

Overview:

Get a record containing the current state of the given VBD_metrics.

Signature:

```
VBD_metrics record get_record (session ref session_id, VBD_metrics ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VBD_metrics record`

all fields from the object

RPC name: get_uuid   This message is removed.

Overview:

Get the uuid field of the given VBD_metrics.

Signature:

```
string get_uuid (session ref session_id, VBD_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config   This message is removed.

Overview:

Remove the given key and its corresponding value from the other_config field of the given VBD_metrics. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VBD_metrics ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_other_config This message is removed.

Overview:

Set the other_config field of the given VBD_metrics.

Signature:

```
void set_other_config (session ref session_id, VBD_metrics ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VBD_metrics ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

Class: VDI

A virtual disk image

Fields for class: VDI

Field	Type	Qualifier	Description
allow_caching	`bool`	*RO/runtime*	true if this VDI is to be cached in the local cache SR
allowed_operations	`vdi_operations set`	*RO/runtime*	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.
cbt_enabled	`bool`	*RO/runtime*	True if changed blocks are tracked for this VDI
crash_dumps	`crashdump ref set`	*RO/runtime*	list of crash dumps that refer to this disk
current_operations	`(string -> vdi_operations) map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
is_a_snapshot	`bool`	*RO/runtime*	true if this is a snapshot.
is_tools_iso	`bool`	*RO/runtime*	Whether this VDI is a Tools ISO
location	`string`	*RO/runtime*	location information
managed	`bool`	*RO/runtime*	

Field	Type	Qualifier	Description
metadata_latest	`bool`	*RO/runtime*	Whether this VDI contains the latest known accessible metadata for the pool
metadata_of_pool	`pool ref`	*RO/runtime*	The pool whose metadata is contained in this VDI
missing	`bool`	*RO/runtime*	true if SR scan operation reported this VDI as not present on disk
name_description	`string`	*RO/constructor*	a notes field containing human-readable description
name_label	`string`	*RO/constructor*	a human-readable name
on_boot	`on_boot`	*RO/runtime*	The behaviour of this VDI on a VM boot
other_config	`(string -> string)map`	*RW*	additional configuration
parent	`VDI ref`	*RO/runtime*	**Deprecated**. This field is always null. Deprecated
physical_utilisation	`int`	*RO/runtime*	amount of physical space that the disk image is currently taking up on the storage repository (in bytes)
read_only	`bool`	*RO/constructor*	true if this disk may ONLY be mounted read-only
sharable	`bool`	*RO/constructor*	true if this disk may be shared
sm_config	`(string -> string)map`	*RW*	SM dependent data

Field	Type	Qualifier	Description
snapshot_of	`VDI ref`	*RO/runtime*	Ref pointing to the VDI this snapshot is of.
snapshot_time	`datetime`	*RO/runtime*	Date/time when this snapshot was created.
snapshots	`VDI ref set`	*RO/runtime*	List pointing to all the VDIs snapshots.
SR	`SR ref`	*RO/constructor*	storage repository in which the VDI resides
storage_lock	`bool`	*RO/runtime*	true if this disk is locked at the storage level
tags	`string set`	*RW*	user-specified tags for categorization purposes
type	`vdi_type`	*RO/constructor*	type of the VDI
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VBDs	`VBD ref set`	*RO/runtime*	list of vbds that refer to this disk
virtual_size	**`int`**	*RO/constructor*	size of disk as presented to the guest (in bytes). Note that, depending on storage backend type, requested size may not be respected exactly
xenstore_data	`(string -> string)map`	*RW*	data to be inserted into the xenstore tree (/local/domain/0/backend/vbd///sm data) after the VDI is attached. This is generally set by the SM backends on vdi_attach.

RPCs associated with class: VDI

RPC name: add_tags *Overview:*

Add the given value to the tags field of the given VDI. If the value is already in that Set, then do nothing.

Signature:

```
void add_tags (session ref session_id, VDI ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	value	New value to add

Minimum Role: vm-operator

Return Type: `void`

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VDI.

Signature:

```
void add_to_other_config (session ref session_id, VDI ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_sm_config *Overview:*

Add the given key-value pair to the sm_config field of the given VDI.

Signature:

```
void add_to_sm_config (session ref session_id, VDI ref self, string key
    , string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_xenstore_data *Overview:*

Add the given key-value pair to the xenstore_data field of the given VDI.

Signature:

```
void add_to_xenstore_data (session ref session_id, VDI ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: clone *Overview:*

Take an exact copy of the VDI and return a reference to the new disk. If any driver_params are specified then these are passed through to the storage-specific substrate driver that implements the clone operation. NB the clone lives in the same Storage Repository as its parent.

Signature:

```
VDI ref clone (session ref session_id, VDI ref vdi, (string -> string)
    map driver_params)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to clone
`(string -> string)map`	driver_params	Optional parameters that are passed through to the backend driver in order to specify storage-type-specific clone options

Minimum Role: vm-admin

Return Type: `VDI ref`

The ID of the newly created VDI.

RPC name: copy *Overview:*

Copy either a full VDI or the block differences between two VDIs into either a fresh VDI or an existing VDI.

Signature:

```
VDI ref copy (session ref session_id, VDI ref vdi, SR ref sr, VDI ref
    base_vdi, VDI ref into_vdi)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to copy
`SR ref`	sr	The destination SR (only required if the destination VDI is not specified
`VDI ref`	base_vdi	The base VDI (only required if copying only changed blocks, by default all blocks will be copied)
`VDI ref`	into_vdi	The destination VDI to copy blocks into (if omitted then a destination SR must be provided and a fresh VDI will be created)

Minimum Role: vm-admin

Return Type: `VDI ref`

The reference of the VDI where the blocks were written.

Possible Error Codes: `VDI_READONLY`, `VDI_TOO_SMALL`, `VDI_NOT_SPARSE`

RPC name: create *Overview:*

Create a new VDI instance, and return its handle.

Signature:

```
VDI ref create (session ref session_id, VDI record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI record`	args	All constructor arguments

Minimum Role: vm-admin

Return Type: `VDI ref`

reference to the newly created object

RPC name: data_destroy *Overview:*

Delete the data of the snapshot VDI, but keep its changed block tracking metadata. When successful, this call changes the type of the VDI to cbt_metadata. This operation is idempotent: calling it on a VDI of type cbt_metadata results in a no-op, and no error will be thrown.

Signature:

```
void data_destroy (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI whose data should be deleted.

Minimum Role: vm-admin

Return Type: `void`

Possible Error Codes: `SR_OPERATION_NOT_SUPPORTED`, `VDI_MISSING`, `SR_NOT_ATTACHED`, `SR_HAS_NO_PBDS`, `OPERATION_NOT_ALLOWED`, `VDI_INCOMPATIBLE_TYPE`, `VDI_NO_CBT_METADATA`, `VDI_IN_USE`, `VDI_IS_A_PHYSICAL_DEVICE`

RPC name: destroy *Overview:*

Destroy the specified VDI instance.

Signature:

```
void destroy (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: disable_cbt *Overview:*

Disable changed block tracking for the VDI. This call is only allowed on VDIs that support enabling CBT. It is an idempotent operation - disabling CBT for a VDI for which CBT is not enabled results in a no-op, and no error will be thrown.

Signature:

```
void disable_cbt (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI for which CBT should be disabled

Minimum Role: vm-admin

Return Type: **`void`**

Possible Error Codes: `SR_OPERATION_NOT_SUPPORTED`, `VDI_MISSING`, `SR_NOT_ATTACHED`, `SR_HAS_NO_PBDS`, `OPERATION_NOT_ALLOWED`, `VDI_INCOMPATIBLE_TYPE`, `VDI_ON_BOOT_MODE_INC`

RPC name: enable_cbt *Overview:*

Enable changed block tracking for the VDI. This call is idempotent - enabling CBT for a VDI for which CBT is already enabled results in a no-op, and no error will be thrown.

Signature:

```
void enable_cbt (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI for which CBT should be enabled

Minimum Role: vm-admin

Return Type: `void`

Possible Error Codes: `SR_OPERATION_NOT_SUPPORTED`, `VDI_MISSING`, `SR_NOT_ATTACHED`, `SR_HAS_NO_PBDS`, `OPERATION_NOT_ALLOWED`, `VDI_INCOMPATIBLE_TYPE`, `VDI_ON_BOOT_MODE_INC`

RPC name: forget *Overview:*

Removes a VDI record from the database

Signature:

```
void forget (session ref session_id, VDI ref vdi)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to forget about

Minimum Role: vm-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the VDIs known to the system.

Signature:

```
VDI ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VDI ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VDI references to VDI records for all VDIs known to the system.

Signature:

```
(VDI ref -> VDI record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VDI ref -> VDI record)map`

records of all objects

RPC name: get_allow_caching *Overview:*

Get the allow_caching field of the given VDI.

Signature:

```
bool get_allow_caching (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VDI.

Signature:

```
vdi_operations set get_allowed_operations (session ref session_id, VDI
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vdi_operations set`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the VDI instances with the given label.

Signature:

```
VDI ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `VDI ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the VDI instance with the specified UUID.

Signature:

```
VDI ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VDI ref`

reference to the object

RPC name: get_cbt_enabled *Overview:*

Get the cbt_enabled field of the given VDI.

Signature:

```
bool get_cbt_enabled (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_crash_dumps *Overview:*

Get the crash_dumps field of the given VDI.

Signature:

```
crashdump ref set get_crash_dumps (session ref session_id, VDI ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `crashdump ref set`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VDI.

Signature:

```
(string -> vdi_operations) map get_current_operations (session ref
    session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vdi_operations)map`

value of the field

RPC name: get_is_a_snapshot *Overview:*

Get the is_a_snapshot field of the given VDI.

Signature:

```
bool get_is_a_snapshot (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_tools_iso *Overview:*

Get the is_tools_iso field of the given VDI.

Signature:

```
bool get_is_tools_iso (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_location *Overview:*

Get the location field of the given VDI.

Signature:

```
string get_location (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_managed *Overview:*

Get the managed field of the given VDI.

Signature:

```
bool get_managed (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_metadata_latest *Overview:*

Get the metadata_latest field of the given VDI.

Signature:

```
bool get_metadata_latest (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_metadata_of_pool *Overview:*

Get the metadata_of_pool field of the given VDI.

Signature:

```
pool ref get_metadata_of_pool (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `pool ref`

value of the field

RPC name: get_missing *Overview:*

Get the missing field of the given VDI.

Signature:

```
bool get_missing (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given VDI.

Signature:

```
string get_name_description (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given VDI.

Signature:

```
string get_name_label (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_nbd_info *Overview:*

Get details specifying how to access this VDI via a Network Block Device server. For each of a set of NBD server addresses on which the VDI is available, the return value set contains a vdi_nbd_server_info object that contains an exportname to request once the NBD connection is established, and connection details for the address. An empty list is returned if there is no network that has a PIF on a host with access to the relevant SR, or if no such network has been assigned an NBD-related purpose in its purpose field. To access the given VDI, any of the vdi_nbd_server_info objects can be used to make a connection to a server, and then the VDI will be available by requesting the exportname.

Signature:

```
vdi_nbd_server_info record set get_nbd_info (session ref session_id,
    VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to access via Network Block Device protocol

Minimum Role: vm-admin

Return Type: `vdi_nbd_server_info record set`

The details necessary for connecting to the VDI over NBD. This includes an authentication token, so must be treated as sensitive material and must not be sent over insecure networks.

Possible Error Codes: `VDI_INCOMPATIBLE_TYPE`

RPC name: get_on_boot *Overview:*

Get the on_boot field of the given VDI.

Signature:

```
on_boot get_on_boot (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `on_boot`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VDI.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VDI
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: (string -> string)map

value of the field

RPC name: get_parent **This message is deprecated.**

Overview:

Get the parent field of the given VDI.

Signature:

```
VDI ref get_parent (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: VDI ref

value of the field

RPC name: get_physical_utilisation *Overview:*

Get the physical_utilisation field of the given VDI.

Signature:

```
int get_physical_utilisation (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`int`**

value of the field

RPC name: get_read_only *Overview:*

Get the read_only field of the given VDI.

Signature:

```
bool get_read_only (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VDI.

Signature:

```
VDI record get_record (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI record`

all fields from the object

RPC name: get_sharable *Overview:*

Get the sharable field of the given VDI.

Signature:

```
bool get_sharable (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_sm_config *Overview:*

Get the sm_config field of the given VDI.

Signature:

```
(string -> string) map get_sm_config (session ref session_id, VDI ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_snapshot_of *Overview:*

Get the snapshot_of field of the given VDI.

Signature:

```
VDI ref get_snapshot_of (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

RPC name: get_snapshot_time *Overview:*

Get the snapshot_time field of the given VDI.

Signature:

```
datetime get_snapshot_time (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_snapshots *Overview:*

Get the snapshots field of the given VDI.

Signature:

```
VDI ref set get_snapshots (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref set`

value of the field

RPC name: get_SR *Overview:*

Get the SR field of the given VDI.

Signature:

```
SR ref get_SR (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_storage_lock *Overview:*

Get the storage_lock field of the given VDI.

Signature:

```
bool get_storage_lock (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_tags *Overview:*

Get the tags field of the given VDI.

Signature:

```
string set get_tags (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given VDI.

Signature:

```
vdi_type get_type (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vdi_type`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VDI.

Signature:

```
string get_uuid (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VBDs *Overview:*

Get the VBDs field of the given VDI.

Signature:

```
VBD ref set get_VBDs (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	self	reference to the object

Minimum Role: read-only

Return Type: `VBD ref set`

value of the field

RPC name: get_virtual_size *Overview:*

Get the virtual_size field of the given VDI.

Signature:

```
int get_virtual_size (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_xenstore_data *Overview:*

Get the xenstore_data field of the given VDI.

Signature:

```
(string -> string) map get_xenstore_data (session ref session_id, VDI
   ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: introduce *Overview:*

Create a new VDI record in the database only

Signature:

```
VDI ref introduce (session ref session_id, string uuid, string
   name_label, string name_description, SR ref SR, vdi_type type, bool
   sharable, bool read_only, (string -> string) map other_config,
   string location, (string -> string) map xenstore_data, (string ->
   string) map sm_config, bool managed, int virtual_size, int
   physical_utilisation, pool ref metadata_of_pool, bool is_a_snapshot,
    datetime snapshot_time, VDI ref snapshot_of)
```

```
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	The uuid of the disk to introduce
`string`	name_label	The name of the disk record
`string`	name_description	The description of the disk record
`SR ref`	SR	The SR that the VDI is in
`vdi_type`	type	The type of the VDI
`bool`	sharable	true if this disk may be shared
`bool`	read_only	true if this disk may ONLY be mounted read-only
`(string -> string)map`	other_config	additional configuration
`string`	location	location information
`(string -> string)map`	xenstore_data	Data to insert into xenstore
`(string -> string)map`	sm_config	Storage-specific config
`bool`	managed	Storage-specific config
`int`	virtual_size	Storage-specific config
`int`	physical_utilisation	Storage-specific config
`pool ref`	metadata_of_pool	Storage-specific config
`bool`	is_a_snapshot	Storage-specific config
`datetime`	snapshot_time	Storage-specific config
`VDI ref`	snapshot_of	Storage-specific config

Minimum Role: vm-admin

Return Type: `VDI ref`

The ref of the newly created VDI record.

Possible Error Codes: `SR_OPERATION_NOT_SUPPORTED`

RPC name: list_changed_blocks *Overview:*

Compare two VDIs in 64k block increments and report which blocks differ. This operation is not allowed when vdi_to is attached to a VM.

Signature:

```
string list_changed_blocks (session ref session_id, VDI ref vdi_from,
    VDI ref vdi_to)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi_from	The first VDI.
`VDI ref`	vdi_to	The second VDI.

Minimum Role: vm-operator

Return Type: `string`

A base64 string-encoding of the bitmap showing which blocks differ in the two VDIs.

Possible Error Codes: `SR_OPERATION_NOT_SUPPORTED`, `VDI_MISSING`, `SR_NOT_ATTACHED`, `SR_HAS_NO_PBDS`, `VDI_IN_USE`

RPC name: open_database *Overview:*

Load the metadata found on the supplied VDI and return a session reference which can be used in API calls to query its contents.

Signature:

```
session ref open_database (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI which contains the database to open

Minimum Role: pool-operator

Return Type: `session ref`

A session which can be used to query the database

RPC name: pool_migrate *Overview:*

Migrate a VDI, which may be attached to a running guest, to a different SR. The destination SR must be visible to the guest.

Signature:

```
VDI ref pool_migrate (session ref session_id, VDI ref vdi, SR ref sr, (
    string -> string) map options)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to migrate
`SR ref`	sr	The destination SR
`(string -> string)map`	options	Other parameters

Minimum Role: vm-power-admin

Return Type: `VDI ref`

The new reference of the migrated VDI.

RPC name: read_database_pool_uuid *Overview:*

Check the VDI cache for the pool UUID of the database on this VDI.

Signature:

```
string read_database_pool_uuid (session ref session_id, VDI ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The metadata VDI to look up in the cache.

Minimum Role: read-only

Return Type: `string`

The cached pool UUID of the database on the VDI.

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VDI. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VDI ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_sm_config *Overview:*

Remove the given key and its corresponding value from the sm_config field of the given VDI. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_sm_config (session ref session_id, VDI ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_xenstore_data *Overview:*

Remove the given key and its corresponding value from the xenstore_data field of the given VDI. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_xenstore_data (session ref session_id, VDI ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_tags *Overview:*

Remove the given value from the tags field of the given VDI. If the value is not in that Set, then do nothing.

Signature:

```
void remove_tags (session ref session_id, VDI ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string`	value	Value to remove

Minimum Role: vm-operator

Return Type: `void`

RPC name: resize *Overview:*

Resize the VDI.

Signature:

```
void resize (session ref session_id, VDI ref vdi, int size)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to resize
`int`	size	The new size of the VDI

Minimum Role: vm-admin

Return Type: `void`

RPC name: resize_online **This message is removed.**

Overview:

Resize the VDI which may or may not be attached to running guests.

Signature:

```
void resize_online (session ref session_id, VDI ref vdi, int size)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to resize
`int`	size	The new size of the VDI

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_allow_caching *Overview:*

Set the value of the allow_caching parameter. This value can only be changed when the VDI is not attached to a running VM. The caching behaviour is only affected by this flag for VHD-based VDIs that have one parent and no child VHDs. Moreover, caching only takes place when the host running the VM containing this VDI has a nominated SR for local caching.

Signature:

```
void set_allow_caching (session ref session_id, VDI ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to modify
`bool`	value	The value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_name_description *Overview:*

Set the name description of the VDI. This can only happen when its SR is currently attached.

Signature:

```
void set_name_description (session ref session_id, VDI ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to modify
`string`	value	The name description for the VDI

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name label of the VDI. This can only happen when then its SR is currently attached.

Signature:

```
void set_name_label (session ref session_id, VDI ref self, string value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to modify
`string`	value	The name lable for the VDI

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_on_boot *Overview:*

Set the value of the on_boot parameter. This value can only be changed when the VDI is not attached to a running VM.

Signature:

```
void set_on_boot (session ref session_id, VDI ref self, on_boot value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to modify
`on_boot`	value	The value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VDI.

Signature:

```
void set_other_config (session ref session_id, VDI ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_read_only *Overview:*

Sets the VDI's read_only field

Signature:

```
void set_read_only (session ref session_id, VDI ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to modify
`bool`	value	The new value of the VDI's read_only field

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_sharable *Overview:*

Sets the VDI's sharable field

Signature:

```
void set_sharable (session ref session_id, VDI ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	The VDI to modify
`bool`	value	The new value of the VDI's sharable field

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_sm_config *Overview:*

Set the sm_config field of the given VDI.

Signature:

```
void set_sm_config (session ref session_id, VDI ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_tags *Overview:*

Set the tags field of the given VDI.

Signature:

```
void set_tags (session ref session_id, VDI ref self, string set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`string set`	value	New value to set

Minimum Role: vm-operator

Return Type: `void`

RPC name: set_xenstore_data *Overview:*

Set the xenstore_data field of the given VDI.

Signature:

```
void set_xenstore_data (session ref session_id, VDI ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: snapshot *Overview:*

Take a read-only snapshot of the VDI, returning a reference to the snapshot. If any driver_params are specified then these are passed through to the storage-specific substrate driver that takes the snapshot. NB the snapshot lives in the same Storage Repository as its parent.

Signature:

```
VDI ref snapshot (session ref session_id, VDI ref vdi, (string ->
    string) map driver_params)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VDI ref`	vdi	The VDI to snapshot

type	name	description
(string -> string)map	driver_params	Optional parameters that can be passed through to backend driver in order to specify storage-type-specific snapshot options

Minimum Role: vm-admin

Return Type: `VDI ref`

The ID of the newly created VDI.

RPC name: update *Overview:*

Ask the storage backend to refresh the fields in the VDI object

Signature:

```
void update (session ref session_id, VDI ref vdi)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VDI ref	vdi	The VDI whose stats (eg size) should be updated

Minimum Role: vm-admin

Return Type: `void`

Possible Error Codes: `SR_OPERATION_NOT_SUPPORTED`

Class: vdi_nbd_server_info

Details for connecting to a VDI using the Network Block Device protocol

Fields for class: vdi_nbd_server_info

Field	Type	Qualifier	Description
address	`string`	*RO/runtime*	An address on which the server can be reached; this can be IPv4, IPv6, or a DNS name.
cert	`string`	*RO/runtime*	The TLS certificate of the server
exportname	`string`	*RO/runtime*	The exportname to request over NBD. This holds details including an authentication token, so it must be protected appropriately. Clients should regard the exportname as an opaque string or token.
port	`int`	*RO/runtime*	The TCP port
subject	`string`	*RO/runtime*	For convenience, this redundant field holds a DNS (hostname) subject of the certificate. This can be a wildcard, but only for a certificate that has a wildcard subject and no concrete hostname subjects.

RPCs associated with class: vdi_nbd_server_info

Class vdi_nbd_server_info has no additional RPCs associated with it.

Class: VGPU

A virtual GPU (vGPU)

Fields for class: VGPU

Field	Type	Qualifier	Description
compatibility_metadata	`(string -> string)map`	*RO/runtime*	VGPU metadata to determine whether a VGPU can migrate between two PGPUs
currently_attached	`bool`	*RO/runtime*	Reflects whether the virtual device is currently connected to a physical device
device	`string`	*RO/runtime*	Order in which the devices are plugged into the VM
extra_args	`string`	*RW*	Extra arguments for vGPU and passed to demu
GPU_group	`GPU_group ref`	*RO/runtime*	GPU group used by the vGPU
other_config	`(string -> string)map`	*RW*	Additional configuration
PCI	`PCI ref`	*RO/runtime*	Device passed trough to VM, either as full device or SR-IOV virtual function
resident_on	`PGPU ref`	*RO/runtime*	The PGPU on which this VGPU is running
scheduled_to_be_resident_on	`PGPU ref`	*RO/runtime*	The PGPU on which this VGPU is scheduled to run
type	`VGPU_type ref`	*RO/runtime*	Preset type for this VGPU
uuid	`string`	*RO/runtime*	Unique identifier/object reference

Field	Type	Qualifier	Description
VM	`VM ref`	*RO/runtime*	VM that owns the vGPU

RPCs associated with class: VGPU

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VGPU.

Signature:

```
void add_to_other_config (session ref session_id, VGPU ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: create *Overview:*

Signature:

```
VGPU ref create (session ref session_id, VM ref VM, GPU_group ref
    GPU_group, string device, (string -> string) map other_config,
    VGPU_type ref type)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	VM	
GPU_group ref	GPU_group	
string	device	
(string -> string)map	other_config	
VGPU_type ref	type	

Minimum Role: pool-operator

Return Type: `VGPU ref`

reference to the newly created object

RPC name: destroy *Overview:*

Signature:

```
void destroy (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VGPU ref	self	The vGPU to destroy

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the VGPUs known to the system.

Signature:

```
VGPU ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VGPU ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VGPU references to VGPU records for all VGPUs known to the system.

Signature:

```
(VGPU ref -> VGPU record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VGPU ref -> VGPU record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the VGPU instance with the specified UUID.

Signature:

```
VGPU ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VGPU ref`

reference to the object

RPC name: get_compatibility_metadata *Overview:*

Get the compatibility_metadata field of the given VGPU.

Signature:

```
(string -> string) map get_compatibility_metadata (session ref
    session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given VGPU.

Signature:

```
bool get_currently_attached (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_device *Overview:*

Get the device field of the given VGPU.

Signature:

```
string get_device (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_extra_args *Overview:*

Get the extra_args field of the given VGPU.

Signature:

```
string get_extra_args (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_GPU_group *Overview:*

Get the GPU_group field of the given VGPU.

Signature:

```
GPU_group ref get_GPU_group (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `GPU_group ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VGPU.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VGPU
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PCI *Overview:*

Get the PCI field of the given VGPU.

Signature:

```
PCI ref get_PCI (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI ref`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VGPU.

Signature:

```
VGPU record get_record (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU record`

all fields from the object

RPC name: get_resident_on *Overview:*

Get the resident_on field of the given VGPU.

Signature:

```
PGPU ref get_resident_on (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU ref`

value of the field

RPC name: get_scheduled_to_be_resident_on *Overview:*

Get the scheduled_to_be_resident_on field of the given VGPU.

Signature:

```
PGPU ref get_scheduled_to_be_resident_on (session ref session_id, VGPU
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU ref`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given VGPU.

Signature:

```
VGPU_type ref get_type (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VGPU.

Signature:

```
string get_uuid (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VM *Overview:*

Get the VM field of the given VGPU.

Signature:

```
VM ref get_VM (session ref session_id, VGPU ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VGPU. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VGPU ref self,
   string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_extra_args *Overview:*

Set the extra_args field of the given VGPU.

Signature:

```
void set_extra_args (session ref session_id, VGPU ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VGPU.

Signature:

```
void set_other_config (session ref session_id, VGPU ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Class: VGPU_type

A type of virtual GPU

Fields for class: VGPU_type

Field	Type	Qualifier	Description
compatible_types_in_vm	`VGPU_type ref set`	*RO/runtime*	List of VGPU types which are compatible in one VM
enabled_on_GPU_groups	`GPU_group ref set`	*RO/runtime*	List of GPU groups in which at least one have this VGPU type enabled
enabled_on_PGPUs	`PGPU ref set`	*RO/runtime*	List of PGPUs that have this VGPU type enabled
experimental	`bool`	*RO/constructor*	Indicates whether VGPUs of this type should be considered experimental
framebuffer_size	`int`	*RO/constructor*	Framebuffer size of the VGPU type, in bytes
identifier	`string`	*RO/constructor*	Key used to identify VGPU types and avoid creating duplicates - this field is used internally and not intended for interpretation by API clients
implementation	`vgpu_type_implemen`	*RO/constructor*	The internal implementation of this VGPU type
max_heads	`int`	*RO/constructor*	Maximum number of displays supported by the VGPU type
max_resolution_x	`int`	*RO/constructor*	Maximum resolution (width) supported by the VGPU type
max_resolution_y	`int`	*RO/constructor*	Maximum resolution (height) supported by the VGPU type

Field	Type	Qualifier	Description
model_name	`string`	*RO/constructor*	Model name associated with the VGPU type
supported_on_GPU_groups	`GPU_group ref set`	*RO/runtime*	List of GPU groups in which at least one PGPU supports this VGPU type
supported_on_PGPUs	`PGPU ref set`	*RO/runtime*	List of PGPUs that support this VGPU type
uuid	`string`	*RO/runtime*	Unique identifier/object reference
vendor_name	`string`	*RO/constructor*	Name of VGPU vendor
VGPUs	`VGPU ref set`	*RO/runtime*	List of VGPUs of this type

RPCs associated with class: VGPU_type

RPC name: get_all *Overview:*

Return a list of all the VGPU_types known to the system.

Signature:

```
VGPU_type ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VGPU_type ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VGPU_type references to VGPU_type records for all VGPU_types known to the system.

Signature:

```
(VGPU_type ref -> VGPU_type record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VGPU_type ref -> VGPU_type record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the VGPU_type instance with the specified UUID.

Signature:

```
VGPU_type ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VGPU_type ref`

reference to the object

RPC name: get_compatible_types_in_vm *Overview:*

Get the compatible_types_in_vm field of the given VGPU_type.

Signature:

```
VGPU_type ref set get_compatible_types_in_vm (session ref session_id,
    VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type ref set`

value of the field

RPC name: get_enabled_on_GPU_groups *Overview:*

Get the enabled_on_GPU_groups field of the given VGPU_type.

Signature:

```
GPU_group ref set get_enabled_on_GPU_groups (session ref session_id,
    VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `GPU_group ref set`

value of the field

RPC name: get_enabled_on_PGPUs *Overview:*

Get the enabled_on_PGPUs field of the given VGPU_type.

Signature:

```
PGPU ref set get_enabled_on_PGPUs (session ref session_id, VGPU_type
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU ref set`

value of the field

RPC name: get_experimental *Overview:*

Get the experimental field of the given VGPU_type.

Signature:

```
bool get_experimental (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_framebuffer_size *Overview:*

Get the framebuffer_size field of the given VGPU_type.

Signature:

```
int get_framebuffer_size (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_identifier *Overview:*

Get the identifier field of the given VGPU_type.

Signature:

```
string get_identifier (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_implementation *Overview:*

Get the implementation field of the given VGPU_type.

Signature:

```
vgpu_type_implementation get_implementation (session ref session_id,
    VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vgpu_type_implementation`

value of the field

RPC name: get_max_heads *Overview:*

Get the max_heads field of the given VGPU_type.

Signature:

```
int get_max_heads (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_max_resolution_x *Overview:*

Get the max_resolution_x field of the given VGPU_type.

Signature:

```
int get_max_resolution_x (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VGPU_type ref	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_max_resolution_y *Overview:*

Get the max_resolution_y field of the given VGPU_type.

Signature:

```
int get_max_resolution_y (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VGPU_type ref	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_model_name *Overview:*

Get the model_name field of the given VGPU_type.

Signature:

```
string get_model_name (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VGPU_type.

Signature:

```
VGPU_type record get_record (session ref session_id, VGPU_type ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU_type record`

all fields from the object

RPC name: get_supported_on_GPU_groups *Overview:*

Get the supported_on_GPU_groups field of the given VGPU_type.

Signature:

```
GPU_group ref set get_supported_on_GPU_groups (session ref session_id,
    VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `GPU_group ref set`

value of the field

RPC name: get_supported_on_PGPUs *Overview:*

Get the supported_on_PGPUs field of the given VGPU_type.

Signature:

```
PGPU ref set get_supported_on_PGPUs (session ref session_id, VGPU_type
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PGPU ref set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VGPU_type.

Signature:

```
string get_uuid (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_vendor_name *Overview:*

Get the vendor_name field of the given VGPU_type.

Signature:

```
string get_vendor_name (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VGPUs *Overview:*

Get the VGPUs field of the given VGPU_type.

Signature:

```
VGPU ref set get_VGPUs (session ref session_id, VGPU_type ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VGPU_type ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU ref set`

value of the field

Class: VIF

A virtual network interface

Fields for class: VIF

Field	Type	Qualifier	Description
allowed_operations	`vif_operations set`	*RO/runtime*	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.
current_operations	`(string -> vif_operations) map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
currently_attached	`bool`	*RO/constructor*	is the device currently attached (erased on reboot)

Field	Type	Qualifier	Description
device	`string`	*RO/constructor*	order in which VIF backends are created by xapi
ipv4_addresses	`string set`	*RO/runtime*	IPv4 addresses in CIDR format
ipv4_allowed	`string set`	*RO/constructor*	A list of IPv4 addresses which can be used to filter traffic passing through this VIF
ipv4_configuration_mode	`vif_ipv4_configuration_mode`	*RO/runtime*	Determines whether IPv4 addresses are configured on the VIF
ipv4_gateway	`string`	*RO/runtime*	IPv4 gateway (the empty string means that no gateway is set)
ipv6_addresses	`string set`	*RO/runtime*	IPv6 addresses in CIDR format
ipv6_allowed	`string set`	*RO/constructor*	A list of IPv6 addresses which can be used to filter traffic passing through this VIF
ipv6_configuration_mode	`vif_ipv6_configuration_mode`	*RO/runtime*	Determines whether IPv6 addresses are configured on the VIF
ipv6_gateway	`string`	*RO/runtime*	IPv6 gateway (the empty string means that no gateway is set)
locking_mode	`vif_locking_mode`	*RO/constructor*	current locking mode of the VIF
MAC	`string`	*RO/constructor*	ethernet MAC address of virtual interface, as exposed to guest
MAC_autogenerated	`bool`	*RO/runtime*	true if the MAC was autogenerated; false indicates it was set manually

Field	Type	Qualifier	Description
metrics	`VIF_metrics ref`	*RO/runtime*	**Removed**. metrics associated with this VIF
MTU	`int`	*RO/constructor*	MTU in octets
network	`network ref`	*RO/constructor*	virtual network to which this vif is connected
other_config	`(string -> string)map`	*RW*	additional configuration
qos_algorithm_params	`(string -> string)map`	*RW*	parameters for chosen QoS algorithm
qos_algorithm_type	`string`	*RW*	QoS algorithm to use
qos_supported_algorithms	`string set`	*RO/runtime*	supported QoS algorithms for this VIF
runtime_properties	`(string -> string)map`	*RO/runtime*	Device runtime properties
status_code	`int`	*RO/runtime*	error/success code associated with last attach-operation (erased on reboot)
status_detail	`string`	*RO/runtime*	error/success information associated with last attach-operation status (erased on reboot)
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VM	`VM ref`	*RO/constructor*	virtual machine to which this vif is connected

RPCs associated with class: VIF

RPC name: add_ipv4_allowed *Overview:*

Associates an IPv4 address with this VIF

Signature:

```
void add_ipv4_allowed (session ref session_id, VIF ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF which the IP address will be associated with
`string`	value	The IP address which will be associated with the VIF

Minimum Role: pool-operator

Return Type: `void`

RPC name: add_ipv6_allowed *Overview:*

Associates an IPv6 address with this VIF

Signature:

```
void add_ipv6_allowed (session ref session_id, VIF ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF which the IP address will be associated with
`string`	value	The IP address which will be associated with the VIF

Minimum Role: pool-operator

Return Type: **void**

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VIF.

Signature:

```
void add_to_other_config (session ref session_id, VIF ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF ref	self	reference to the object
string	key	Key to add
string	value	Value to add

Minimum Role: vm-admin

Return Type: **void**

RPC name: add_to_qos_algorithm_params *Overview:*

Add the given key-value pair to the qos/algorithm_params field of the given VIF.

Signature:

```
void add_to_qos_algorithm_params (session ref session_id, VIF ref self,
     string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF ref	self	reference to the object
string	key	Key to add

type	name	description
string	value	Value to add

Minimum Role: vm-admin

Return Type: **void**

RPC name: configure_ipv4 *Overview:*

Configure IPv4 settings for this virtual interface

Signature:

```
void configure_ipv4 (session ref session_id, VIF ref self,
    vif_ipv4_configuration_mode mode, string address, string gateway)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF ref	self	The VIF to configure
vif_ipv4_configuration_mode	mode	Whether to use static or no IPv4 assignment
string	address	The IPv4 address in / format (for static mode only)
string	gateway	The IPv4 gateway (for static mode only; leave empty to not set a gateway)

Minimum Role: vm-operator

Return Type: **void**

RPC name: configure_ipv6 *Overview:*

Configure IPv6 settings for this virtual interface

Signature:

```
void configure_ipv6 (session ref session_id, VIF ref self,
    vif_ipv6_configuration_mode mode, string address, string gateway)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF to configure
`vif_ipv6_configuration_mode`	mode	Whether to use static or no IPv6 assignment
`string`	address	The IPv6 address in / format (for static mode only)
`string`	gateway	The IPv6 gateway (for static mode only; leave empty to not set a gateway)

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: create *Overview:*

Create a new VIF instance, and return its handle.

Signature:

```
VIF ref create (session ref session_id, VIF record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF record`	args	All constructor arguments

Minimum Role: vm-admin

Return Type: `VIF ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified VIF instance.

Signature:

```
void destroy (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: vm-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the VIFs known to the system.

Signature:

```
VIF ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VIF ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VIF references to VIF records for all VIFs known to the system.

Signature:

```
(VIF ref -> VIF record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VIF ref -> VIF record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VIF.

Signature:

```
vif_operations set get_allowed_operations (session ref session_id, VIF
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vif_operations set`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the VIF instance with the specified UUID.

Signature:

```
VIF ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VIF ref`

reference to the object

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VIF.

Signature:

```
(string -> vif_operations) map get_current_operations (session ref
    session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vif_operations)map`

value of the field

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given VIF.

Signature:

```
bool get_currently_attached (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_device *Overview:*

Get the device field of the given VIF.

Signature:

```
string get_device (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_ipv4_addresses *Overview:*

Get the ipv4_addresses field of the given VIF.

Signature:

```
string set get_ipv4_addresses (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_ipv4_allowed *Overview:*

Get the ipv4_allowed field of the given VIF.

Signature:

```
string set get_ipv4_allowed (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_ipv4_configuration_mode *Overview:*

Get the ipv4_configuration_mode field of the given VIF.

Signature:

```
vif_ipv4_configuration_mode get_ipv4_configuration_mode (session ref
    session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vif_ipv4_configuration_mode`

value of the field

RPC name: get_ipv4_gateway *Overview:*

Get the ipv4_gateway field of the given VIF.

Signature:

```
string get_ipv4_gateway (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_ipv6_addresses *Overview:*

Get the ipv6_addresses field of the given VIF.

Signature:

```
string set get_ipv6_addresses (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_ipv6_allowed *Overview:*

Get the ipv6_allowed field of the given VIF.

Signature:

```
string set get_ipv6_allowed (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_ipv6_configuration_mode *Overview:*

Get the ipv6_configuration_mode field of the given VIF.

Signature:

```
vif_ipv6_configuration_mode get_ipv6_configuration_mode (session ref
    session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vif_ipv6_configuration_mode`

value of the field

RPC name: get_ipv6_gateway *Overview:*

Get the ipv6_gateway field of the given VIF.

Signature:

```
string get_ipv6_gateway (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_locking_mode *Overview:*

Get the locking_mode field of the given VIF.

Signature:

```
vif_locking_mode get_locking_mode (session ref session_id, VIF ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vif_locking_mode`

value of the field

RPC name: get_MAC *Overview:*

Get the MAC field of the given VIF.

Signature:

```
string get_MAC (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_MAC_autogenerated *Overview:*

Get the MAC_autogenerated field of the given VIF.

Signature:

```
bool get_MAC_autogenerated (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_metrics **This message is removed.**

Overview:

Get the metrics field of the given VIF.

Signature:

```
VIF_metrics ref get_metrics (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF ref	self	reference to the object

Minimum Role: read-only

Return Type: `VIF_metrics ref`

value of the field

RPC name: get_MTU *Overview:*

Get the MTU field of the given VIF.

Signature:

```
int get_MTU (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF ref	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_network *Overview:*

Get the network field of the given VIF.

Signature:

```
network ref get_network (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `network ref`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VIF.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VIF
   ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_qos_algorithm_params *Overview:*

Get the qos/algorithm_params field of the given VIF.

Signature:

```
(string -> string) map get_qos_algorithm_params (session ref session_id
    , VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_qos_algorithm_type *Overview:*

Get the qos/algorithm_type field of the given VIF.

Signature:

```
string get_qos_algorithm_type (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_qos_supported_algorithms *Overview:*

Get the qos/supported_algorithms field of the given VIF.

Signature:

```
string set get_qos_supported_algorithms (session ref session_id, VIF
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VIF.

Signature:

```
VIF record get_record (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VIF record`

all fields from the object

RPC name: get_runtime_properties *Overview:*

Get the runtime_properties field of the given VIF.

Signature:

```
(string -> string) map get_runtime_properties (session ref session_id,
    VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_status_code *Overview:*

Get the status_code field of the given VIF.

Signature:

```
int get_status_code (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_status_detail *Overview:*

Get the status_detail field of the given VIF.

Signature:

```
string get_status_detail (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VIF.

Signature:

```
string get_uuid (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VM *Overview:*

Get the VM field of the given VIF.

Signature:

```
VM ref get_VM (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: move *Overview:*

Move the specified VIF to the specified network, even while the VM is running

Signature:

```
void move (session ref session_id, VIF ref self, network ref network)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF to move
`network ref`	network	The network to move it to

Minimum Role: vm-admin

Return Type: `void`

RPC name: plug *Overview:*

Hotplug the specified VIF, dynamically attaching it to the running VM

Signature:

```
void plug (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF to hotplug

Minimum Role: vm-admin

Return Type: `void`

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VIF. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VIF ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: `void`

RPC name: remove_from_qos_algorithm_params *Overview:*

Remove the given key and its corresponding value from the qos/algorithm_params field of the given VIF. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_qos_algorithm_params (session ref session_id, VIF ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: `void`

RPC name: remove_ipv4_allowed *Overview:*

Removes an IPv4 address from this VIF

Signature:

```
void remove_ipv4_allowed (session ref session_id, VIF ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF from which the IP address will be removed
`string`	value	The IP address which will be removed from the VIF

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_ipv6_allowed *Overview:*

Removes an IPv6 address from this VIF

Signature:

```
void remove_ipv6_allowed (session ref session_id, VIF ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF from which the IP address will be removed
`string`	value	The IP address which will be removed from the VIF

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_ipv4_allowed *Overview:*

Set the IPv4 addresses to which traffic on this VIF can be restricted

Signature:

```
void set_ipv4_allowed (session ref session_id, VIF ref self, string set
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF which the IP addresses will be associated with
`string set`	value	The IP addresses which will be associated with the VIF

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_ipv6_allowed *Overview:*

Set the IPv6 addresses to which traffic on this VIF can be restricted

Signature:

```
void set_ipv6_allowed (session ref session_id, VIF ref self, string set
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF which the IP addresses will be associated with
`string set`	value	The IP addresses which will be associated with the VIF

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_locking_mode *Overview:*

Set the locking mode for this VIF

Signature:

```
void set_locking_mode (session ref session_id, VIF ref self,
   vif_locking_mode value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF whose locking mode will be set
`vif_locking_mode`	value	The new locking mode for the VIF

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VIF.

Signature:

```
void set_other_config (session ref session_id, VIF ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_qos_algorithm_params *Overview:*

Set the qos/algorithm_params field of the given VIF.

Signature:

```
void set_qos_algorithm_params (session ref session_id, VIF ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_qos_algorithm_type *Overview:*

Set the qos/algorithm_type field of the given VIF.

Signature:

```
void set_qos_algorithm_type (session ref session_id, VIF ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: unplug *Overview:*

Hot-unplug the specified VIF, dynamically unattaching it from the running VM

Signature:

```
void unplug (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF to hot-unplug

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: unplug_force *Overview:*

Forcibly unplug the specified VIF

Signature:

```
void unplug_force (session ref session_id, VIF ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF ref`	self	The VIF to forcibly unplug

Minimum Role: vm-admin

Return Type: `void`

Class: VIF_metrics

This class is removed.

The metrics associated with a virtual network device

Fields for class: VIF_metrics

Field	Type	Qualifier	Description
io_read_kbs	`float`	*RO/runtime*	**Removed**. Read bandwidth (KiB/s)
io_write_kbs	`float`	*RO/runtime*	**Removed**. Write bandwidth (KiB/s)
last_updated	`datetime`	*RO/runtime*	**Removed**. Time at which this information was last updated
other_config	`(string -> string)map`	*RW*	**Removed**. additional configuration

Field	Type	Qualifier	Description
uuid	`string`	*RO/runtime*	**Removed**. Unique identifier/object reference

RPCs associated with class: VIF_metrics

RPC name: add_to_other_config   This message is removed.

Overview:

Add the given key-value pair to the other_config field of the given VIF_metrics.

Signature:

```
void add_to_other_config (session ref session_id, VIF_metrics ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: get_all   This message is removed.

Overview:

Return a list of all the VIF_metrics instances known to the system.

Signature:

```
VIF_metrics ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VIF_metrics ref set`

references to all objects

RPC name: get_all_records This message is removed.

Overview:

Return a map of VIF_metrics references to VIF_metrics records for all VIF_metrics instances known to the system.

Signature:

```
(VIF_metrics ref -> VIF_metrics record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VIF_metrics ref -> VIF_metrics record)map`

records of all objects

RPC name: get_by_uuid This message is removed.

Overview:

Get a reference to the VIF_metrics instance with the specified UUID.

Signature:

```
VIF_metrics ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VIF_metrics ref`

reference to the object

RPC name: get_io_read_kbs   This message is removed.

Overview:

Get the io/read_kbs field of the given VIF_metrics.

Signature:

```
float get_io_read_kbs (session ref session_id, VIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`float`**

value of the field

RPC name: get_io_write_kbs   This message is removed.

Overview:

Get the io/write_kbs field of the given VIF_metrics.

Signature:

```
float get_io_write_kbs (session ref session_id, VIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`float`**

value of the field

RPC name: get_last_updated This message is removed.

Overview:

Get the last_updated field of the given VIF_metrics.

Signature:

```
datetime get_last_updated (session ref session_id, VIF_metrics ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_other_config This message is removed.

Overview:

Get the other_config field of the given VIF_metrics.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    VIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record This message is removed.

Overview:

Get a record containing the current state of the given VIF_metrics.

Signature:

```
VIF_metrics record get_record (session ref session_id, VIF_metrics ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VIF_metrics record`

all fields from the object

RPC name: get_uuid This message is removed.

Overview:

Get the uuid field of the given VIF_metrics.

Signature:

```
string get_uuid (session ref session_id, VIF_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VIF_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config   This message is removed.

Overview:

Remove the given key and its corresponding value from the other_config field of the given VIF_metrics. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VIF_metrics ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF_metrics ref	self	reference to the object
string	key	Key to remove

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_other_config   This message is removed.

Overview:

Set the other_config field of the given VIF_metrics.

Signature:

```
void set_other_config (session ref session_id, VIF_metrics ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VIF_metrics ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: vm-admin

Return Type: **void**

Class: VLAN

A VLAN mux/demux

Fields for class: VLAN

Field	Type	Qualifier	Description
other_config	(string -> string)map	*RW*	additional configuration
tag	int	*RO/constructor*	VLAN tag in use
tagged_PIF	PIF ref	*RO/constructor*	interface on which traffic is tagged
untagged_PIF	PIF ref	*RO/runtime*	interface on which traffic is untagged
uuid	string	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: VLAN

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VLAN.

Signature:

```
void add_to_other_config (session ref session_id, VLAN ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VLAN ref	self	reference to the object

type	name	description
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-operator

Return Type: `void`

RPC name: create *Overview:*

Create a VLAN mux/demuxer

Signature:

```
VLAN ref create (session ref session_id, PIF ref tagged_PIF, int tag,
    network ref network)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`PIF ref`	tagged_PIF	PIF which receives the tagged traffic
`int`	tag	VLAN tag to use
`network ref`	network	Network to receive the untagged traffic

Minimum Role: pool-operator

Return Type: `VLAN ref`

The reference of the created VLAN object

RPC name: destroy *Overview:*

Destroy a VLAN mux/demuxer

Signature:

```
void destroy (session ref session_id, VLAN ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	VLAN mux/demuxer to destroy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the VLANs known to the system.

Signature:

```
VLAN ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VLAN ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VLAN references to VLAN records for all VLANs known to the system.

Signature:

```
(VLAN ref -> VLAN record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VLAN ref -> VLAN record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the VLAN instance with the specified UUID.

Signature:

```
VLAN ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VLAN ref`

reference to the object

RPC name: get_other_config *Overview:*

Get the other_config field of the given VLAN.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VLAN
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VLAN.

Signature:

```
VLAN record get_record (session ref session_id, VLAN ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VLAN record`

all fields from the object

RPC name: get_tag *Overview:*

Get the tag field of the given VLAN.

Signature:

```
int get_tag (session ref session_id, VLAN ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_tagged_PIF *Overview:*

Get the tagged_PIF field of the given VLAN.

Signature:

```
PIF ref get_tagged_PIF (session ref session_id, VLAN ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_untagged_PIF *Overview:*

Get the untagged_PIF field of the given VLAN.

Signature:

```
PIF ref get_untagged_PIF (session ref session_id, VLAN ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PIF ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VLAN.

Signature:

```
string get_uuid (session ref session_id, VLAN ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VLAN. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VLAN ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given VLAN.

Signature:

```
void set_other_config (session ref session_id, VLAN ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VLAN ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

Class: VM

A virtual machine (or 'guest').

Fields for class: VM

Field	Type	Qualifier	Description
actions_after_crash	`on_crash_behaviour`	*RO/constructor*	action to take if the guest crashes
actions_after_reboot	`on_normal_exit`	*RW*	action to take after the guest has rebooted itself
actions_after_shutdown	`on_normal_exit`	*RW*	action to take after the guest has shutdown itself
actions_after_softreboot	`on_softreboot_behavior`	*RW*	action to take after soft reboot

Field	Type	Qualifier	Description
affinity	`host ref`	*RW*	A host which the VM has some affinity for (or NULL). This is used as a hint to the start call when it decides where to run the VM. Resource constraints may cause the VM to be started elsewhere.
allowed_operations	`vm_operations set`	*RO/runtime*	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.
appliance	`VM_appliance ref`	*RO/constructor*	the appliance to which this VM belongs
attached_PCIs	`PCI ref set`	*RO/runtime*	Currently passed-through PCI devices
bios_strings	`(string -> string)map`	*RO/runtime*	BIOS strings
blobs	`(string -> blob ref)map`	*RO/runtime*	Binary blobs associated with this VM
blocked_operations	`(vm_operations -> string)map`	*RW*	List of operations which have been explicitly blocked and an error code
children	`VM ref set`	*RO/runtime*	List pointing to all the children of this VM
consoles	`console ref set`	*RO/runtime*	virtual console devices
crash_dumps	`crashdump ref set`	*RO/runtime*	crash dumps associated with this VM

Field	Type	Qualifier	Description
current_operations	`(string -> vm_operations) map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
domain_type	`domain_type`	*RO/constructor*	The type of domain that will be created when the VM is started
domarch	`string`	*RO/runtime*	Domain architecture (if available, null string otherwise)
domid	`int`	*RO/runtime*	domain ID (if available, -1 otherwise)
generation_id	`string`	*RO/constructor*	Generation ID of the VM
guest_metrics	`VM_guest_metrics ref`	*RO/runtime*	metrics associated with the running guest
ha_always_run	`bool`	*RO/constructor*	**Deprecated**. if true then the system will attempt to keep the VM running as much as possible.
ha_restart_priority	`string`	*RO/constructor*	has possible values: "best-effort"meaning "try to restart this VM if possible but don't consider the Pool to be overcommitted if this is not possible"; "restart"meaning "this VM should be restarted"; ""meaning "do not try to restart this VM"

Field	Type	Qualifier	Description
hardware_platform_version	`int`	*RW*	The host virtual hardware platform version the VM can run on
has_vendor_device	`bool`	*RO/constructor*	When an HVM guest starts, this controls the presence of the emulated C000 PCI device which triggers Windows Update to fetch or update PV drivers.
HVM_boot_params	`(string -> string) map`	*RW*	HVM boot params
HVM_boot_policy	`string`	*RO/constructor*	**Deprecated**. HVM boot policy
HVM_shadow_multiplier	`float`	*RO/constructor*	multiplier applied to the amount of shadow that will be made available to the guest
is_a_snapshot	`bool`	*RO/runtime*	true if this is a snapshot. Snapshotted VMs can never be started, they are used only for cloning other VMs
is_a_template	`bool`	*RW*	true if this is a template. Template VMs can never be started, they are used only for cloning other VMs
is_control_domain	`bool`	*RO/runtime*	true if this is a control domain (domain 0 or a driver domain)

Field	Type	Qualifier	Description
is_default_template	`bool`	*RO/runtime*	true if this is a default template. Default template VMs can never be started or migrated, they are used only for cloning other VMs
is_snapshot_from_vmpp	`bool`	*RO/constructor*	**Removed**. true if this snapshot was created by the protection policy
is_vmss_snapshot	`bool`	*RO/constructor*	true if this snapshot was created by the snapshot schedule
last_boot_CPU_flags	`(string -> string) map`	*RO/constructor*	describes the CPU flags on which the VM was last booted
last_booted_record	`string`	*RO/constructor*	marshalled value containing VM record at time of last boot
memory_dynamic_max	`int`	*RO/constructor*	Dynamic maximum (bytes)
memory_dynamic_min	`int`	*RO/constructor*	Dynamic minimum (bytes)
memory_overhead	`int`	*RO/runtime*	Virtualization memory overhead (bytes).
memory_static_max	`int`	*RO/constructor*	Statically-set (i.e. absolute) maximum (bytes). The value of this field at VM start time acts as a hard limit of the amount of memory a guest can use. New values only take effect on reboot.

Field	Type	Qualifier	Description
memory_static_min	`int`	*RO/constructor*	Statically-set (i.e. absolute) mininum (bytes). The value of this field indicates the least amount of memory this VM can boot with without crashing.
memory_target	`int`	*RO/constructor*	**Deprecated**. Dynamically-set memory target (bytes). The value of this field indicates the current target for memory available to this VM.
metrics	`VM_metrics ref`	*RO/runtime*	metrics associated with this VM
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
NVRAM	`(string -> string) map`	*RO/constructor*	initial value for guest NVRAM (containing UEFI variables, etc). Cannot be changed while the VM is running
order	`int`	*RO/constructor*	The point in the startup or shutdown sequence at which this VM will be started
other_config	`(string -> string) map`	*RW*	additional configuration
parent	`VM ref`	*RO/runtime*	Ref pointing to the parent of this VM

Field	Type	Qualifier	Description
PCI_bus	`string`	*RW*	**Deprecated**. PCI bus path for pass-through devices
pending_guidances	`update_guidances set`	*RO/runtime*	The set of pending guidances after applying updates
platform	`(string -> string)map`	*RW*	platform-specific configuration
power_state	`vm_power_state`	*RO/constructor*	Current power state of the machine
protection_policy	`VMPP ref`	*RO/constructor*	**Deprecated**. Ref pointing to a protection policy for this VM
PV_args	`string`	*RW*	kernel command-line arguments
PV_bootloader	`string`	*RW*	name of or path to bootloader
PV_bootloader_args	`string`	*RW*	miscellaneous arguments for the bootloader
PV_kernel	`string`	*RW*	path to the kernel
PV_legacy_args	`string`	*RW*	to make Zurich guests boot
PV_ramdisk	`string`	*RW*	path to the initrd
recommendations	`string`	*RW*	An XML specification of recommended values and ranges for properties of this VM

Field	Type	Qualifier	Description
reference_label	`string`	*RO/constructor*	Textual reference to the template used to create a VM. This can be used by clients in need of an immutable reference to the template since the latter's uuid and name_label may change, for example, after a package installation or upgrade.
requires_reboot	`bool`	*RO/runtime*	Indicates whether a VM requires a reboot in order to update its configuration, e.g. its memory allocation.
resident_on	`host ref`	*RO/runtime*	the host the VM is currently resident on
scheduled_to_be_resident_on	`host ref`	*RO/runtime*	the host on which the VM is due to be started/resumed/migrated. This acts as a memory reservation indicator
shutdown_delay	`int`	*RO/constructor*	The delay to wait before proceeding to the next order in the shutdown sequence (seconds)
snapshot_info	`(string -> string)map`	*RO/runtime*	Human-readable information concerning this snapshot

Field	Type	Qualifier	Description
snapshot_metadata	`string`	*RO/runtime*	Encoded information about the VM's metadata this is a snapshot of
snapshot_of	`VM ref`	*RO/runtime*	Ref pointing to the VM this snapshot is of.
snapshot_schedule	`VMSS ref`	*RO/constructor*	Ref pointing to a snapshot schedule for this VM
snapshot_time	`datetime`	*RO/runtime*	Date/time when this snapshot was created.
snapshots	`VM ref set`	*RO/runtime*	List pointing to all the VM snapshots.
start_delay	`int`	*RO/constructor*	The delay to wait before proceeding to the next order in the startup sequence (seconds)
suspend_SR	`SR ref`	*RW*	The SR on which a suspend image is stored
suspend_VDI	`VDI ref`	*RO/constructor*	The VDI that a suspend image is stored on. (Only has meaning if VM is currently suspended)
tags	`string set`	*RW*	user-specified tags for categorization purposes
transportable_snapshot_id	`string`	*RO/runtime*	Transportable ID of the snapshot VM
user_version	`int`	*RW*	Creators of VMs and templates may store version information here.

Field	Type	Qualifier	Description
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VBDs	`VBD ref set`	*RO/runtime*	virtual block devices
VCPUs_at_startup	`int`	*RO/constructor*	Boot number of VCPUs
VCPUs_max	`int`	*RO/constructor*	Max number of VCPUs
VCPUs_params	`(string -> string)map`	*RW*	configuration parameters for the selected VCPU policy
version	`int`	*RO/constructor*	The number of times this VM has been recovered
VGPUs	`VGPU ref set`	*RO/runtime*	Virtual GPUs
VIFs	`VIF ref set`	*RO/runtime*	virtual network interfaces
VTPMs	`VTPM ref set`	*RO/runtime*	virtual TPMs
VUSBs	`VUSB ref set`	*RO/runtime*	vitual usb devices
xenstore_data	`(string -> string)map`	*RW*	data to be inserted into the xenstore tree (/local/domain//vm-data) after the VM is created.

RPCs associated with class: VM

RPC name: add_tags *Overview:*

Add the given value to the tags field of the given VM. If the value is already in that Set, then do nothing.

Signature:

```
void add_tags (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to add

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: add_to_blocked_operations *Overview:*

Add the given key-value pair to the blocked_operations field of the given VM.

Signature:

```
void add_to_blocked_operations (session ref session_id, VM ref self,
    vm_operations key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`vm_operations`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: add_to_HVM_boot_params *Overview:*

Add the given key-value pair to the HVM/boot_params field of the given VM.

Signature:

```
void add_to_HVM_boot_params (session ref session_id, VM ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_NVRAM *Overview:*

Signature:

```
void add_to_NVRAM (session ref session_id, VM ref self, string key,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	key	The key
`string`	value	The value

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VM.

Signature:

```
void add_to_other_config (session ref session_id, VM ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: add_to_platform *Overview:*

Add the given key-value pair to the platform field of the given VM.

Signature:

```
void add_to_platform (session ref session_id, VM ref self, string key,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: add_to_VCPUs_params *Overview:*

Add the given key-value pair to the VCPUs/params field of the given VM.

Signature:

```
void add_to_VCPUs_params (session ref session_id, VM ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_VCPUs_params_live *Overview:*

Add the given key-value pair to VM.VCPUs_params, and apply that value on the running VM

Signature:

```
void add_to_VCPUs_params_live (session ref session_id, VM ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	key	The key
`string`	value	The value

Minimum Role: vm-admin

Return Type: `void`

RPC name: add_to_xenstore_data *Overview:*

Add the given key-value pair to the xenstore_data field of the given VM.

Signature:

```
void add_to_xenstore_data (session ref session_id, VM ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: assert_agile *Overview:*

Returns an error if the VM is not considered agile e.g. because it is tied to a resource local to a host

Signature:

```
void assert_agile (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM

Minimum Role: read-only

Return Type: **`void`**

RPC name: assert_can_be_recovered *Overview:*

Assert whether all SRs required to recover this VM are available.

Signature:

```
void assert_can_be_recovered (session ref session_id, VM ref self,
    session ref session_to)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to recover
`session ref`	session_to	The session to which the VM is to be recovered.

Minimum Role: read-only

Return Type: `void`

Possible Error Codes: `VM_IS_PART_OF_AN_APPLIANCE`, `VM_REQUIRES_SR`

RPC name: assert_can_boot_here *Overview:*

Returns an error if the VM could not boot on this host for some reason

Signature:

```
void assert_can_boot_here (session ref session_id, VM ref self, host
    ref host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`host ref`	host	The host

Minimum Role: read-only

Return Type: `void`

Possible Error Codes: `HOST_NOT_ENOUGH_FREE_MEMORY`, `HOST_NOT_ENOUGH_PCPUS`, `NETWORK_SRIOV_INSUFFICIENT_CAPACITY`, `HOST_NOT_LIVE`, `HOST_DISABLED`, `HOST_CANNOT_ATTACH_NETWORK`, `VM_HVM_REQUIRED`, `VM_REQUIRES_GPU`, `VM_REQUIRES_IOMMU`, `VM_REQUIRES_NETWORK`, `VM_REQUIRES_SR`, `VM_REQUIRES_VGPU`, `VM_HOST_INCOMPATIBLE_VERSI`, `VM_HOST_INCOMPATIBLE_VIRTUAL_HARDWARE_PLATFORM_VERSION`, `INVALID_VALUE`, `MEMORY_CONSTRAINT_VIOLATION`, `OPERATION_NOT_ALLOWED`, `VALUE_NOT_SUPPORTED`, `VM_INCOMPATIBLE_WITH_THIS_HOST`

RPC name: assert_can_migrate *Overview:*

Assert whether a VM can be migrated to the specified destination.

Signature:

```
void assert_can_migrate (session ref session_id, VM ref vm, (string ->
    string) map dest, bool live, (VDI ref -> SR ref) map vdi_map, (VIF
    ref -> network ref) map vif_map, (string -> string) map options, (
    VGPU ref -> GPU_group ref) map vgpu_map)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM
`(string -> string)map`	dest	The result of a VM.migrate_receive call.
`bool`	live	Live migration
`(VDI ref -> SR ref) map`	vdi_map	Map of source VDI to destination SR
`(VIF ref -> network ref)map`	vif_map	Map of source VIF to destination network
`(string -> string)map`	options	Other parameters
`(VGPU ref -> GPU_group ref)map`	vgpu_map	Map of source vGPU to destination GPU group

Minimum Role: vm-power-admin

Return Type: `void`

Possible Error Codes: `LICENCE_RESTRICTION`

RPC name: assert_operation_valid *Overview:*

Check to see whether this operation is acceptable in the current state of the system, raising an error if the operation is invalid for some reason

Signature:

```
void assert_operation_valid (session ref session_id, VM ref self,
    vm_operations op)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`vm_operations`	op	proposed operation

Minimum Role: read-only

Return Type: **`void`**

RPC name: call_plugin *Overview:*

Call an API plugin on this vm

Signature:

```
string call_plugin (session ref session_id, VM ref vm, string plugin,
    string fn, (string -> string) map args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The vm
`string`	plugin	The name of the plugin

type	name	description
`string`	fn	The name of the function within the plugin
`(string -> string)map`	args	Arguments for the function

Minimum Role: vm-operator

Return Type: `string`

Result from the plugin

RPC name: checkpoint *Overview:*

Checkpoints the specified VM, making a new VM. Checkpoint automatically exploits the capabilities of the underlying storage repository in which the VM's disk images are stored (e.g. Copy on Write) and saves the memory image as well.

Signature:

```
VM ref checkpoint (session ref session_id, VM ref vm, string new_name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to be checkpointed
`string`	new_name	The name of the checkpointed VM

Minimum Role: vm-power-admin

Return Type: `VM ref`

The reference of the newly created VM.

Possible Error Codes: `VM_BAD_POWER_STATE`, `SR_FULL`, `OPERATION_NOT_ALLOWED`, `VM_CHECKPOINT_SUSPEND_FAILED`, `VM_CHECKPOINT_RESUME_FAILED`

RPC name: clean_reboot *Overview:*

Attempt to cleanly shutdown the specified VM (Note: this may not be supported---e.g. if a guest agent is not installed). This can only be called when the specified VM is in the Running state.

Signature:

```
void clean_reboot (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to shutdown

Minimum Role: vm-operator

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALL`
, `VM_IS_TEMPLATE`

RPC name: clean_shutdown *Overview:*

Attempt to cleanly shutdown the specified VM. (Note: this may not be supported---e.g. if a guest agent is not installed). This can only be called when the specified VM is in the Running state.

Signature:

```
void clean_shutdown (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to shutdown

Minimum Role: vm-operator

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALL`
, `VM_IS_TEMPLATE`

RPC name: clone *Overview:*

Clones the specified VM, making a new VM. Clone automatically exploits the capabilities of the underlying storage repository in which the VM's disk images are stored (e.g. Copy on Write). This function can only be called when the VM is in the Halted State.

Signature:

```
VM ref clone (session ref session_id, VM ref vm, string new_name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to be cloned
`string`	new_name	The name of the cloned VM

Minimum Role: vm-admin

Return Type: `VM ref`

The reference of the newly created VM.

Possible Error Codes: `VM_BAD_POWER_STATE`, `SR_FULL`, `OPERATION_NOT_ALLOWED`, `LICENCE_RESTRICTION`

RPC name: compute_memory_overhead *Overview:*

Computes the virtualization memory overhead of a VM.

Signature:

```
int compute_memory_overhead (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM for which to compute the memory overhead

Minimum Role: read-only

Return Type: `int`

the virtualization memory overhead of the VM.

RPC name: copy *Overview:*

Copied the specified VM, making a new VM. Unlike clone, copy does not exploits the capabilities of the underlying storage repository in which the VM's disk images are stored. Instead, copy guarantees that the disk images of the newly created VM will be 'full disks'- i.e. not part of a CoW chain. This function can only be called when the VM is in the Halted State.

Signature:

```
VM ref copy (session ref session_id, VM ref vm, string new_name, SR ref
      sr)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to be copied
`string`	new_name	The name of the copied VM
`SR ref`	sr	An SR to copy all the VM's disks into (if an invalid reference then it uses the existing SRs)

Minimum Role: vm-admin

Return Type: `VM ref`

The reference of the newly created VM.

Possible Error Codes: `VM_BAD_POWER_STATE`, `SR_FULL`, `OPERATION_NOT_ALLOWED`, `LICENCE_RESTRICTION`

RPC name: copy_bios_strings *Overview:*

Copy the BIOS strings from the given host to this VM

Signature:

```
void copy_bios_strings (session ref session_id, VM ref vm, host ref
    host)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to modify
`host ref`	host	The host to copy the BIOS strings from

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: create *Overview:*

NOT RECOMMENDED! VM.clone or VM.copy (or VM.import) is a better choice in almost all situations. The standard way to obtain a new VM is to call VM.clone on a template VM, then call VM.provision on the new clone. Caution: if VM.create is used and then the new VM is attached to a virtual disc that has an operating system already installed, then there is no guarantee that the operating system will boot and run. Any software that calls VM.create on a future version of this API may fail or give unexpected results. For example this could happen if an additional parameter were added to VM.create. VM.create is intended only for use in the automatic creation of the system VM templates. It creates a new VM instance, and returns its handle.

Signature:

```
VM ref create (session ref session_id, VM record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM record`	args	All constructor arguments

Minimum Role: vm-admin

Return Type: `VM ref`

reference to the newly created object

RPC name: create_new_blob *Overview:*

Create a placeholder for a named binary blob of data that is associated with this VM

Signature:

```
blob ref create_new_blob (session ref session_id, VM ref vm, string
    name, string mime_type, bool public)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM
`string`	name	The name associated with the blob
`string`	mime_type	The mime type for the data. Empty string translates to application/octet-stream
`bool`	public	True if the blob should be publicly available

Minimum Role: vm-power-admin

Return Type: `blob ref`

The reference of the blob, needed for populating its data

RPC name: destroy *Overview:*

Destroy the specified VM. The VM is completely removed from the system. This function can only be called when the VM is in the Halted State.

Signature:

```
void destroy (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: vm-admin

Return Type: `void`

RPC name: forget_data_source_archives *Overview:*

Forget the recorded statistics related to the specified data source

Signature:

```
void forget_data_source_archives (session ref session_id, VM ref self,
    string data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	data_source	The data source whose archives are to be forgotten

Minimum Role: vm-admin

Return Type: `void`

RPC name: get_actions_after_crash *Overview:*

Get the actions/after_crash field of the given VM.

Signature:

```
on_crash_behaviour get_actions_after_crash (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `on_crash_behaviour`

value of the field

RPC name: get_actions_after_reboot *Overview:*

Get the actions/after_reboot field of the given VM.

Signature:

```
on_normal_exit get_actions_after_reboot (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `on_normal_exit`

value of the field

RPC name: get_actions_after_shutdown *Overview:*

Get the actions/after_shutdown field of the given VM.

Signature:

```
on_normal_exit get_actions_after_shutdown (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `on_normal_exit`

value of the field

RPC name: get_actions_after_softreboot *Overview:*

Get the actions/after_softreboot field of the given VM.

Signature:

```
on_softreboot_behavior get_actions_after_softreboot (session ref
    session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `on_softreboot_behavior`

value of the field

RPC name: get_affinity *Overview:*

Get the affinity field of the given VM.

Signature:

```
host ref get_affinity (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_all *Overview:*

Return a list of all the VMs known to the system.

Signature:

```
VM ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VM ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VM references to VM records for all VMs known to the system.

Signature:

```
(VM ref -> VM record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VM ref -> VM record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VM.

Signature:

```
vm_operations set get_allowed_operations (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vm_operations set`

value of the field

RPC name: get_allowed_VBD_devices *Overview:*

Returns a list of the allowed values that a VBD device field can take

Signature:

```
string set get_allowed_VBD_devices (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to query

Minimum Role: read-only

Return Type: `string set`

The allowed values

RPC name: get_allowed_VIF_devices *Overview:*

Returns a list of the allowed values that a VIF device field can take

Signature:

```
string set get_allowed_VIF_devices (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to query

Minimum Role: read-only

Return Type: `string set`

The allowed values

RPC name: get_appliance *Overview:*

Get the appliance field of the given VM.

Signature:

```
VM_appliance ref get_appliance (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM_appliance ref`

value of the field

RPC name: get_attached_PCIs *Overview:*

Get the attached_PCIs field of the given VM.

Signature:

```
PCI ref set get_attached_PCIs (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `PCI ref set`

value of the field

RPC name: get_bios_strings *Overview:*

Get the bios_strings field of the given VM.

Signature:

```
(string -> string) map get_bios_strings (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_blobs *Overview:*

Get the blobs field of the given VM.

Signature:

```
(string -> blob ref) map get_blobs (session ref session_id, VM ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> blob ref)map`

value of the field

RPC name: get_blocked_operations *Overview:*

Get the blocked_operations field of the given VM.

Signature:

```
(vm_operations -> string) map get_blocked_operations (session ref
    session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(vm_operations -> string)map`

value of the field

RPC name: get_boot_record **This message is deprecated.**

Overview:

Returns a record describing the VM's dynamic state, initialised when the VM boots and updated to reflect runtime configuration changes e.g. CPU hotplug

Signature:

```
VM record get_boot_record (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM whose boot-time state to return

Minimum Role: read-only

Return Type: `VM record`

A record describing the VM

RPC name: get_by_name_label *Overview:*

Get all the VM instances with the given label.

Signature:

```
VM ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `VM ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the VM instance with the specified UUID.

Signature:

```
VM ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VM ref`

reference to the object

RPC name: get_children *Overview:*

Get the children field of the given VM.

Signature:

```
VM ref set get_children (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref set`

value of the field

RPC name: get_consoles *Overview:*

Get the consoles field of the given VM.

Signature:

```
console ref set get_consoles (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `console ref set`

value of the field

RPC name: get_cooperative **This message is deprecated.**

Overview:

Return true if the VM is currently 'co-operative'i.e. is expected to reach a balloon target and actually has done

Signature:

```
bool get_cooperative (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM

Minimum Role: read-only

Return Type: `bool`

true if the VM is currently 'co-operative'; false otherwise

RPC name: get_crash_dumps *Overview:*

Get the crash_dumps field of the given VM.

Signature:

```
crashdump ref set get_crash_dumps (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `crashdump ref set`

value of the field

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VM.

Signature:

```
(string -> vm_operations) map get_current_operations (session ref
    session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vm_operations)map`

value of the field

RPC name: get_data_sources *Overview:*

Signature:

```
data_source record set get_data_sources (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to interrogate

Minimum Role: read-only

Return Type: `data_source record set`

A set of data sources

RPC name: get_domain_type *Overview:*

Get the domain_type field of the given VM.

Signature:

```
domain_type get_domain_type (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `domain_type`

value of the field

RPC name: get_domarch *Overview:*

Get the domarch field of the given VM.

Signature:

```
string get_domarch (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_domid *Overview:*

Get the domid field of the given VM.

Signature:

```
int get_domid (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_generation_id *Overview:*

Get the generation_id field of the given VM.

Signature:

```
string get_generation_id (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_guest_metrics *Overview:*

Get the guest_metrics field of the given VM.

Signature:

```
VM_guest_metrics ref get_guest_metrics (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM_guest_metrics ref`

value of the field

RPC name: get_ha_always_run **This message is deprecated.**

Overview:

Get the ha_always_run field of the given VM.

Signature:

```
bool get_ha_always_run (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_ha_restart_priority *Overview:*

Get the ha_restart_priority field of the given VM.

Signature:

```
string get_ha_restart_priority (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_hardware_platform_version *Overview:*

Get the hardware_platform_version field of the given VM.

Signature:

```
int get_hardware_platform_version (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`int`**

value of the field

RPC name: get_has_vendor_device *Overview:*

Get the has_vendor_device field of the given VM.

Signature:

```
bool get_has_vendor_device (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_HVM_boot_params *Overview:*

Get the HVM/boot_params field of the given VM.

Signature:

```
(string -> string) map get_HVM_boot_params (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_HVM_boot_policy **This message is deprecated.**

Overview:

Get the HVM/boot_policy field of the given VM.

Signature:

```
string get_HVM_boot_policy (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_HVM_shadow_multiplier *Overview:*

Get the HVM/shadow_multiplier field of the given VM.

Signature:

```
float get_HVM_shadow_multiplier (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: **`float`**

value of the field

RPC name: get_is_a_snapshot *Overview:*

Get the is_a_snapshot field of the given VM.

Signature:

```
bool get_is_a_snapshot (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_a_template *Overview:*

Get the is_a_template field of the given VM.

Signature:

```
bool get_is_a_template (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_control_domain *Overview:*

Get the is_control_domain field of the given VM.

Signature:

```
bool get_is_control_domain (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_default_template *Overview:*

Get the is_default_template field of the given VM.

Signature:

```
bool get_is_default_template (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_snapshot_from_vmpp **This message is removed.**

Overview:

Get the is_snapshot_from_vmpp field of the given VM.

Signature:

```
bool get_is_snapshot_from_vmpp (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_vmss_snapshot *Overview:*

Get the is_vmss_snapshot field of the given VM.

Signature:

```
bool get_is_vmss_snapshot (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_last_boot_CPU_flags *Overview:*

Get the last_boot_CPU_flags field of the given VM.

Signature:

```
(string -> string) map get_last_boot_CPU_flags (session ref session_id,
    VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_last_booted_record *Overview:*

Get the last_booted_record field of the given VM.

Signature:

```
string get_last_booted_record (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_memory_dynamic_max *Overview:*

Get the memory/dynamic_max field of the given VM.

Signature:

```
int get_memory_dynamic_max (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_memory_dynamic_min *Overview:*

Get the memory/dynamic_min field of the given VM.

Signature:

```
int get_memory_dynamic_min (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_memory_overhead *Overview:*

Get the memory/overhead field of the given VM.

Signature:

```
int get_memory_overhead (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_memory_static_max *Overview:*

Get the memory/static_max field of the given VM.

Signature:

```
int get_memory_static_max (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_memory_static_min *Overview:*

Get the memory/static_min field of the given VM.

Signature:

```
int get_memory_static_min (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_memory_target **This message is deprecated.**

Overview:

Get the memory/target field of the given VM.

Signature:

```
int get_memory_target (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_metrics *Overview:*

Get the metrics field of the given VM.

Signature:

```
VM_metrics ref get_metrics (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM_metrics ref`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given VM.

Signature:

```
string get_name_description (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given VM.

Signature:

```
string get_name_label (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_NVRAM *Overview:*

Get the NVRAM field of the given VM.

Signature:

```
(string -> string) map get_NVRAM (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_order *Overview:*

Get the order field of the given VM.

Signature:

```
int get_order (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VM.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_parent *Overview:*

Get the parent field of the given VM.

Signature:

```
VM ref get_parent (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: get_PCI_bus **This message is deprecated.**

Overview:

Get the PCI_bus field of the given VM.

Signature:

```
string get_PCI_bus (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_pending_guidances *Overview:*

Get the pending_guidances field of the given VM.

Signature:

```
update_guidances set get_pending_guidances (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `update_guidances set`

value of the field

RPC name: get_platform *Overview:*

Get the platform field of the given VM.

Signature:

```
(string -> string) map get_platform (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_possible_hosts *Overview:*

Return the list of hosts on which this VM may run.

Signature:

```
host ref set get_possible_hosts (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM

Minimum Role: read-only

Return Type: `host ref set`

The possible hosts

RPC name: get_power_state *Overview:*

Get the power_state field of the given VM.

Signature:

```
vm_power_state get_power_state (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vm_power_state`

value of the field

RPC name: get_protection_policy **This message is deprecated.**

Overview:

Get the protection_policy field of the given VM.

Signature:

```
VMPP ref get_protection_policy (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VMPP ref`

value of the field

RPC name: get_PV_args *Overview:*

Get the PV/args field of the given VM.

Signature:

```
string get_PV_args (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_PV_bootloader *Overview:*

Get the PV/bootloader field of the given VM.

Signature:

```
string get_PV_bootloader (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_PV_bootloader_args *Overview:*

Get the PV/bootloader_args field of the given VM.

Signature:

```
string get_PV_bootloader_args (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_PV_kernel *Overview:*

Get the PV/kernel field of the given VM.

Signature:

```
string get_PV_kernel (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_PV_legacy_args *Overview:*

Get the PV/legacy_args field of the given VM.

Signature:

```
string get_PV_legacy_args (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_PV_ramdisk *Overview:*

Get the PV/ramdisk field of the given VM.

Signature:

```
string get_PV_ramdisk (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_recommendations *Overview:*

Get the recommendations field of the given VM.

Signature:

```
string get_recommendations (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VM.

Signature:

```
VM record get_record (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	reference to the object

Minimum Role: read-only

Return Type: `VM record`

all fields from the object

RPC name: get_reference_label *Overview:*

Get the reference_label field of the given VM.

Signature:

```
string get_reference_label (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_requires_reboot *Overview:*

Get the requires_reboot field of the given VM.

Signature:

```
bool get_requires_reboot (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_resident_on *Overview:*

Get the resident_on field of the given VM.

Signature:

```
host ref get_resident_on (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_scheduled_to_be_resident_on *Overview:*

Get the scheduled_to_be_resident_on field of the given VM.

Signature:

```
host ref get_scheduled_to_be_resident_on (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `host ref`

value of the field

RPC name: get_shutdown_delay *Overview:*

Get the shutdown_delay field of the given VM.

Signature:

```
int get_shutdown_delay (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_snapshot_info *Overview:*

Get the snapshot_info field of the given VM.

Signature:

```
(string -> string) map get_snapshot_info (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_snapshot_metadata *Overview:*

Get the snapshot_metadata field of the given VM.

Signature:

```
string get_snapshot_metadata (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_snapshot_of *Overview:*

Get the snapshot_of field of the given VM.

Signature:

```
VM ref get_snapshot_of (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: get_snapshot_schedule *Overview:*

Get the snapshot_schedule field of the given VM.

Signature:

```
VMSS ref get_snapshot_schedule (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VMSS ref`

value of the field

RPC name: get_snapshot_time *Overview:*

Get the snapshot_time field of the given VM.

Signature:

```
datetime get_snapshot_time (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_snapshots *Overview:*

Get the snapshots field of the given VM.

Signature:

```
VM ref set get_snapshots (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref set`

value of the field

RPC name: get_SRs_required_for_recovery *Overview:*

List all the SR's that are required for the VM to be recovered

Signature:

```
SR ref set get_SRs_required_for_recovery (session ref session_id, VM
    ref self, session ref session_to)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM for which the SRs have to be recovered
`session ref`	session_to	The session to which the SRs of the VM have to be recovered.

Minimum Role: read-only

Return Type: `SR ref set`

refs for SRs required to recover the VM

RPC name: get_start_delay *Overview:*

Get the start_delay field of the given VM.

Signature:

```
int get_start_delay (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_suspend_SR *Overview:*

Get the suspend_SR field of the given VM.

Signature:

```
SR ref get_suspend_SR (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `SR ref`

value of the field

RPC name: get_suspend_VDI *Overview:*

Get the suspend_VDI field of the given VM.

Signature:

```
VDI ref get_suspend_VDI (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VDI ref`

value of the field

RPC name: get_tags *Overview:*

Get the tags field of the given VM.

Signature:

```
string set get_tags (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_transportable_snapshot_id *Overview:*

Get the transportable_snapshot_id field of the given VM.

Signature:

```
string get_transportable_snapshot_id (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_user_version *Overview:*

Get the user_version field of the given VM.

Signature:

```
int get_user_version (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VM.

Signature:

```
string get_uuid (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VBDs *Overview:*

Get the VBDs field of the given VM.

Signature:

```
VBD ref set get_VBDs (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VBD ref set`

value of the field

RPC name: get_VCPUs_at_startup *Overview:*

Get the VCPUs/at_startup field of the given VM.

Signature:

```
int get_VCPUs_at_startup (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_VCPUs_max *Overview:*

Get the VCPUs/max field of the given VM.

Signature:

```
int get_VCPUs_max (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_VCPUs_params *Overview:*

Get the VCPUs/params field of the given VM.

Signature:

```
(string -> string) map get_VCPUs_params (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_version *Overview:*

Get the version field of the given VM.

Signature:

```
int get_version (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_VGPUs *Overview:*

Get the VGPUs field of the given VM.

Signature:

```
VGPU ref set get_VGPUs (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VGPU ref set`

value of the field

RPC name: get_VIFs *Overview:*

Get the VIFs field of the given VM.

Signature:

```
VIF ref set get_VIFs (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VIF ref set`

value of the field

RPC name: get_VTPMs *Overview:*

Get the VTPMs field of the given VM.

Signature:

```
VTPM ref set get_VTPMs (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VTPM ref set`

value of the field

RPC name: get_VUSBs *Overview:*

Get the VUSBs field of the given VM.

Signature:

```
VUSB ref set get_VUSBs (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VUSB ref set`

value of the field

RPC name: get_xenstore_data *Overview:*

Get the xenstore_data field of the given VM.

Signature:

```
(string -> string) map get_xenstore_data (session ref session_id, VM
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: hard_reboot *Overview:*

Stop executing the specified VM without attempting a clean shutdown and immediately restart the VM.

Signature:

```
void hard_reboot (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to reboot

Minimum Role: vm-operator

Return Type: `void`

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALL`
, `VM_IS_TEMPLATE`

RPC name: hard_shutdown *Overview:*

Stop executing the specified VM without attempting a clean shutdown.

Signature:

```
void hard_shutdown (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to destroy

Minimum Role: vm-operator

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALL`
, `VM_IS_TEMPLATE`

RPC name: import *Overview:*

Import an XVA from a URI

Signature:

```
VM ref set import (session ref session_id, string url, SR ref sr, bool
    full_restore, bool force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	url	The URL of the XVA file
`SR ref`	sr	The destination SR for the disks
`bool`	full_restore	Perform a full restore
`bool`	force	Force the import

Minimum Role: pool-operator

Return Type: `VM ref set`

Imported VM reference

RPC name: import_convert *Overview:*

Import using a conversion service.

Signature:

```
void import_convert (session ref session_id, string type, string
    username, string password, SR ref sr, (string -> string) map
    remote_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	type	Type of the conversion
`string`	username	Admin username on the host
`string`	password	Password on the host
`SR ref`	sr	The destination SR
`(string -> string)map`	remote_config	Remote configuration options

Minimum Role: vm-admin

Return Type: `void`

RPC name: maximise_memory *Overview:*

Returns the maximum amount of guest memory which will fit, together with overheads, in the supplied amount of physical memory. If 'exact'is true then an exact calculation is performed using the VM's current settings. If 'exact'is false then a more conservative approximation is used

Signature:

```
int maximise_memory (session ref session_id, VM ref self, int total,
    bool approximate)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	total	Total amount of physical RAM to fit within
`bool`	approximate	If false the limit is calculated with the guest's current exact configuration. Otherwise a more approximate calculation is performed

Minimum Role: read-only

Return Type: `int`

The maximum possible static-max

RPC name: migrate_send *Overview:*

Migrate the VM to another host. This can only be called when the specified VM is in the Running state.

Signature:

```
VM ref migrate_send (session ref session_id, VM ref vm, (string ->
    string) map dest, bool live, (VDI ref -> SR ref) map vdi_map, (VIF
    ref -> network ref) map vif_map, (string -> string) map options, (
    VGPU ref -> GPU_group ref) map vgpu_map)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM
`(string -> string)map`	dest	The result of a Host.migrate_receive call.
`bool`	live	Live migration
`(VDI ref -> SR ref) map`	vdi_map	Map of source VDI to destination SR

type	name	description
(VIF ref -> network ref)map	vif_map	Map of source VIF to destination network
(string -> string)map	options	Other parameters
(VGPU ref -> GPU_group ref)map	vgpu_map	Map of source vGPU to destination GPU group

Minimum Role: vm-power-admin

Return Type: `VM ref`

The reference of the newly created VM in the destination pool

Possible Error Codes: `VM_BAD_POWER_STATE`, `LICENCE_RESTRICTION`

RPC name: pause *Overview:*

Pause the specified VM. This can only be called when the specified VM is in the Running state.

Signature:

```
void pause (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	vm	The VM to pause

Minimum Role: vm-operator

Return Type: `void`

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALL`
, `VM_IS_TEMPLATE`

RPC name: pool_migrate *Overview:*

Migrate a VM to another Host.

Signature:

```
void pool_migrate (session ref session_id, VM ref vm, host ref host, (
    string -> string) map options)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to migrate
`host ref`	host	The target host
`(string -> string)map`	options	Extra configuration operations: force, live, copy, compress. Each is a boolean option, taking 'true' or 'false' as a value. Option 'compress' controls the use of stream compression during migration.

Minimum Role: client-cert

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `VM_IS_TEMPLATE`, `OPERATION_NOT_ALLOWED`, `VM_BAD_POWER_STATE`

RPC name: power_state_reset *Overview:*

Reset the power-state of the VM to halted in the database only. (Used to recover from slave failures in pooling scenarios by resetting the power-states of VMs running on dead slaves to halted.) This is a potentially dangerous operation; use with care.

Signature:

```
void power_state_reset (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`VM ref`	vm	The VM to reset

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: provision *Overview:*

Inspects the disk configuration contained within the VM's other_config, creates VDIs and VBDs and then executes any applicable post-install script.

Signature:

```
void provision (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to be provisioned

Minimum Role: vm-admin

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `SR_FULL`, `OPERATION_NOT_ALLOWED`, `LICENCE_RESTRICTION`

RPC name: query_data_source *Overview:*

Query the latest value of the specified data source

Signature:

```
float query_data_source (session ref session_id, VM ref self, string
    data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	data_source	The data source to query

Minimum Role: read-only

Return Type: **`float`**

The latest value, averaged over the last 5 seconds

RPC name: query_services *Overview:*

Query the system services advertised by this VM and register them. This can only be applied to a system domain.

Signature:

```
(string -> string) map query_services (session ref session_id, VM ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM

Minimum Role: pool-admin

Return Type: `(string -> string)map`

map of service type to name

RPC name: record_data_source *Overview:*

Start recording the specified data source

Signature:

```
void record_data_source (session ref session_id, VM ref self, string
    data_source)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	data_source	The data source to record

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: recover *Overview:*

Recover the VM

Signature:

```
void recover (session ref session_id, VM ref self, session ref
    session_to, bool force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to recover
`session ref`	session_to	The session to which the VM is to be recovered.
`bool`	force	Whether the VM should replace newer versions of itself.

Minimum Role: read-only

Return Type: **`void`**

RPC name: remove_from_blocked_operations *Overview:*

Remove the given key and its corresponding value from the blocked_operations field of the given VM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_blocked_operations (session ref session_id, VM ref
    self, vm_operations key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`vm_operations`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_HVM_boot_params *Overview:*

Remove the given key and its corresponding value from the HVM/boot_params field of the given VM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_HVM_boot_params (session ref session_id, VM ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_NVRAM *Overview:*

Signature:

```
void remove_from_NVRAM (session ref session_id, VM ref self, string key
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	key	The key

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VM ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_platform *Overview:*

Remove the given key and its corresponding value from the platform field of the given VM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_platform (session ref session_id, VM ref self, string
    key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_VCPUs_params *Overview:*

Remove the given key and its corresponding value from the VCPUs/params field of the given VM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_VCPUs_params (session ref session_id, VM ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_from_xenstore_data *Overview:*

Remove the given key and its corresponding value from the xenstore_data field of the given VM. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_xenstore_data (session ref session_id, VM ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: remove_tags *Overview:*

Remove the given value from the tags field of the given VM. If the value is not in that Set, then do nothing.

Signature:

```
void remove_tags (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	Value to remove

Minimum Role: vm-operator

Return Type: **`void`**

RPC name: restart_device_models *Overview:*

Signature:

```
void restart_device_models (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM

Minimum Role: client-cert

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OTHER_OPERATION_IN_PROGRESS`, `VM_IS_TEMPLATE`, `OPERATION_NOT_ALLOWED`, `VM_BAD_POWER_STATE`

RPC name: resume *Overview:*

Awaken the specified VM and resume it. This can only be called when the specified VM is in the Suspended state.

Signature:

```
void resume (session ref session_id, VM ref vm, bool start_paused, bool
    force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to resume
`bool`	start_paused	Resume VM in paused state if set to true.

type	name	description
bool	force	Attempt to force the VM to resume. If this flag is false then the VM may fail pre-resume safety checks (e.g. if the CPU the VM was running on looks substantially different to the current one)

Minimum Role: vm-operator

Return Type: **void**

Possible Error Codes: VM_BAD_POWER_STATE, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

RPC name: resume_on *Overview:*

Awaken the specified VM and resume it on a particular Host. This can only be called when the specified VM is in the Suspended state.

Signature:

```
void resume_on (session ref session_id, VM ref vm, host ref host, bool
    start_paused, bool force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	vm	The VM to resume
host ref	host	The Host on which to resume the VM
bool	start_paused	Resume VM in paused state if set to true.

type	name	description
bool	force	Attempt to force the VM to resume. If this flag is false then the VM may fail pre-resume safety checks (e.g. if the CPU the VM was running on looks substantially different to the current one)

Minimum Role: client-cert

Return Type: **void**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OPERATION_NOT_ALLOWED`, `VM_IS_TEMPLATE`

RPC name: retrieve_wlb_recommendations *Overview:*

Returns mapping of hosts to ratings, indicating the suitability of starting the VM at that location according to wlb. Rating is replaced with an error if the VM cannot boot there.

Signature:

```
(host ref -> string set) map retrieve_wlb_recommendations (session ref
    session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	vm	The VM

Minimum Role: read-only

Return Type: `(host ref -> string set)map`

The potential hosts and their corresponding recommendations or errors

RPC name: revert *Overview:*

Reverts the specified VM to a previous state.

Signature:

```
void revert (session ref session_id, VM ref snapshot)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	snapshot	The snapshotted state that we revert to

Minimum Role: vm-power-admin

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `OPERATION_NOT_ALLOWED`, `SR_FULL`, `VM_REVERT_FAILED`

RPC name: send_sysrq *Overview:*

Send the given key as a sysrq to this VM. The key is specified as a single character (a String of length 1). This can only be called when the specified VM is in the Running state.

Signature:

```
void send_sysrq (session ref session_id, VM ref vm, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM
`string`	key	The key to send

Minimum Role: pool-admin

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`

RPC name: send_trigger *Overview:*

Send the named trigger to this VM. This can only be called when the specified VM is in the Running state.

Signature:

```
void send_trigger (session ref session_id, VM ref vm, string trigger)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM
`string`	trigger	The trigger to send

Minimum Role: pool-admin

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`

RPC name: set_actions_after_crash *Overview:*

Sets the actions_after_crash parameter

Signature:

```
void set_actions_after_crash (session ref session_id, VM ref self,
    on_crash_behaviour value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to set
`on_crash_behaviour`	value	The new value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_actions_after_reboot *Overview:*

Set the actions/after_reboot field of the given VM.

Signature:

```
void set_actions_after_reboot (session ref session_id, VM ref self,
    on_normal_exit value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`on_normal_exit`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_actions_after_shutdown *Overview:*

Set the actions/after_shutdown field of the given VM.

Signature:

```
void set_actions_after_shutdown (session ref session_id, VM ref self,
    on_normal_exit value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`on_normal_exit`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_actions_after_softreboot *Overview:*

Set the actions/after_softreboot field of the given VM.

Signature:

```
void set_actions_after_softreboot (session ref session_id, VM ref self,
    on_softreboot_behavior value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`on_softreboot_behavior`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_affinity *Overview:*

Set the affinity field of the given VM.

Signature:

```
void set_affinity (session ref session_id, VM ref self, host ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`host ref`	value	New value to set

Minimum Role: vm-power-admin

Return Type: `void`

RPC name: set_appliance *Overview:*

Assign this VM to an appliance.

Signature:

```
void set_appliance (session ref session_id, VM ref self, VM_appliance
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to assign to an appliance.
`VM_appliance ref`	value	The appliance to which this VM should be assigned.

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_bios_strings *Overview:*

Set custom BIOS strings to this VM. VM will be given a default set of BIOS strings, only some of which can be overridden by the supplied values. Allowed keys are: 'bios-vendor', 'bios-version', 'system-manufacturer', 'system-product-name', 'system-version', 'system-serial-number', 'enclosure-asset-tag', 'baseboard-manufacturer', 'baseboard-product-name', 'baseboard-version', 'baseboard-serial-number', 'baseboard-asset-tag', 'baseboard-location-in-chassis', 'enclosure-asset-tag'

Signature:

```
void set_bios_strings (session ref session_id, VM ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to modify

type	name	description
`(string -> string)map`	value	The custom BIOS strings as a list of key-value pairs

Minimum Role: vm-admin

Return Type: **`void`**

Possible Error Codes: `VM_BIOS_STRINGS_ALREADY_SET`, `INVALID_VALUE`

RPC name: set_blocked_operations *Overview:*

Set the blocked_operations field of the given VM.

Signature:

```
void set_blocked_operations (session ref session_id, VM ref self, (
    vm_operations -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`(vm_operations -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_domain_type *Overview:*

Set the VM.domain_type field of the given VM, which will take effect when it is next started

Signature:

```
void set_domain_type (session ref session_id, VM ref self, domain_type
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`domain_type`	value	The new domain type

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_ha_always_run **This message is deprecated.**

Overview:

Set the value of the ha_always_run

Signature:

```
void set_ha_always_run (session ref session_id, VM ref self, bool value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`bool`	value	The value

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_ha_restart_priority *Overview:*

Set the value of the ha_restart_priority field

Signature:

```
void set_ha_restart_priority (session ref session_id, VM ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	value	The value

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_hardware_platform_version *Overview:*

Set the hardware_platform_version field of the given VM.

Signature:

```
void set_hardware_platform_version (session ref session_id, VM ref self
    , int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`int`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_has_vendor_device *Overview:*

Controls whether, when the VM starts in HVM mode, its virtual hardware will include the emulated PCI device for which drivers may be available through Windows Update. Usually this should never be changed on a VM on which Windows has been installed: changing it on such a VM is likely to lead to a crash on next start.

Signature:

```
void set_has_vendor_device (session ref session_id, VM ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM on which to set this flag
`bool`	value	True to provide the vendor PCI device.

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_HVM_boot_params *Overview:*

Set the HVM/boot_params field of the given VM.

Signature:

```
void set_HVM_boot_params (session ref session_id, VM ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_HVM_boot_policy **This message is deprecated.**

Overview:

Set the VM.HVM_boot_policy field of the given VM, which will take effect when it is next started

Signature:

```
void set_HVM_boot_policy (session ref session_id, VM ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`string`	value	The new HVM boot policy

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_HVM_shadow_multiplier *Overview:*

Set the shadow memory multiplier on a halted VM

Signature:

```
void set_HVM_shadow_multiplier (session ref session_id, VM ref self,
    float value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`float`	value	The new shadow memory multiplier to set

Minimum Role: vm-power-admin

Return Type: `void`

RPC name: set_is_a_template *Overview:*

Set the is_a_template field of the given VM.

Signature:

```
void set_is_a_template (session ref session_id, VM ref self, bool value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_memory *Overview:*

Set the memory allocation of this VM. Sets all of memory_static_max, memory_dynamic_min, and memory_dynamic_max to the given value, and leaves memory_static_min untouched.

Signature:

```
void set_memory (session ref session_id, VM ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	value	The new memory allocation (bytes).

Minimum Role: vm-power-admin

Return Type: **`void`**

RPC name: set_memory_dynamic_max *Overview:*

Set the value of the memory_dynamic_max field

Signature:

```
void set_memory_dynamic_max (session ref session_id, VM ref self, int
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to modify
`int`	value	The new value of memory_dynamic_max

Minimum Role: vm-power-admin

Return Type: `void`

RPC name: set_memory_dynamic_min *Overview:*

Set the value of the memory_dynamic_min field

Signature:

```
void set_memory_dynamic_min (session ref session_id, VM ref self, int
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to modify
`int`	value	The new value of memory_dynamic_min

Minimum Role: vm-power-admin

Return Type: `void`

RPC name: set_memory_dynamic_range *Overview:*

Set the minimum and maximum amounts of physical memory the VM is allowed to use.

Signature:

```
void set_memory_dynamic_range (session ref session_id, VM ref self, int
    min, int max)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	min	The new minimum value
`int`	max	The new maximum value

Minimum Role: vm-power-admin

Return Type: `void`

RPC name: set_memory_limits *Overview:*

Set the memory limits of this VM.

Signature:

```
void set_memory_limits (session ref session_id, VM ref self, int
    static_min, int static_max, int dynamic_min, int dynamic_max)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	static_min	The new value of memory_static_min.
`int`	static_max	The new value of memory_static_max.

type	name	description
`int`	dynamic_min	The new value of memory_dynamic_min.
`int`	dynamic_max	The new value of memory_dynamic_max.

Minimum Role: vm-power-admin

Return Type: **`void`**

RPC name: set_memory_static_max *Overview:*

Set the value of the memory_static_max field

Signature:

```
void set_memory_static_max (session ref session_id, VM ref self, int
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM to modify
`int`	value	The new value of memory_static_max

Minimum Role: vm-power-admin

Return Type: **`void`**

Possible Error Codes: `HA_OPERATION_WOULD_BREAK_FAILOVER_PLAN`

RPC name: set_memory_static_min *Overview:*

Set the value of the memory_static_min field

Signature:

```
void set_memory_static_min (session ref session_id, VM ref self, int
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	The VM to modify
int	value	The new value of memory_static_min

Minimum Role: vm-power-admin

Return Type: **void**

RPC name: set_memory_static_range *Overview:*

Set the static (ie boot-time) range of virtual memory that the VM is allowed to use.

Signature:

```
void set_memory_static_range (session ref session_id, VM ref self, int
    min, int max)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	The VM
int	min	The new minimum value
int	max	The new maximum value

Minimum Role: vm-power-admin

Return Type: **void**

RPC name: set_memory_target_live **This message is deprecated.**

Overview:

Set the memory target for a running VM

Signature:

```
void set_memory_target_live (session ref session_id, VM ref self, int
    target)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	target	The target in bytes

Minimum Role: vm-power-admin

Return Type: `void`

RPC name: set_name_description *Overview:*

Set the name/description field of the given VM.

Signature:

```
void set_name_description (session ref session_id, VM ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_name_label *Overview:*

Set the name/label field of the given VM.

Signature:

```
void set_name_label (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_NVRAM *Overview:*

Signature:

```
void set_NVRAM (session ref session_id, VM ref self, (string -> string)
     map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`(string -> string)map`	value	The value

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_order *Overview:*

Set this VM's boot order

Signature:

```
void set_order (session ref session_id, VM ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	value	This VM's boot order

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_other_config *Overview:*

Set the other_config field of the given VM.

Signature:

```
void set_other_config (session ref session_id, VM ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_PCI_bus **This message is deprecated.**

Overview:

Set the PCI_bus field of the given VM.

Signature:

```
void set_PCI_bus (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_platform *Overview:*

Set the platform field of the given VM.

Signature:

```
void set_platform (session ref session_id, VM ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_protection_policy **This message is removed.**

Overview:

Set the value of the protection_policy field

Signature:

```
void set_protection_policy (session ref session_id, VM ref self, VMPP
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`VMPP ref`	value	The value

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_PV_args *Overview:*

Set the PV/args field of the given VM.

Signature:

```
void set_PV_args (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_PV_bootloader *Overview:*

Set the PV/bootloader field of the given VM.

Signature:

```
void set_PV_bootloader (session ref session_id, VM ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_PV_bootloader_args *Overview:*

Set the PV/bootloader_args field of the given VM.

Signature:

```
void set_PV_bootloader_args (session ref session_id, VM ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_PV_kernel *Overview:*

Set the PV/kernel field of the given VM.

Signature:

```
void set_PV_kernel (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_PV_legacy_args *Overview:*

Set the PV/legacy_args field of the given VM.

Signature:

```
void set_PV_legacy_args (session ref session_id, VM ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_PV_ramdisk *Overview:*

Set the PV/ramdisk field of the given VM.

Signature:

```
void set_PV_ramdisk (session ref session_id, VM ref self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_recommendations *Overview:*

Set the recommendations field of the given VM.

Signature:

```
void set_recommendations (session ref session_id, VM ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_shadow_multiplier_live *Overview:*

Set the shadow memory multiplier on a running VM

Signature:

```
void set_shadow_multiplier_live (session ref session_id, VM ref self,
    float multiplier)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`float`	multiplier	The new shadow memory multiplier to set

Minimum Role: vm-power-admin

Return Type: **`void`**

RPC name: set_shutdown_delay *Overview:*

Set this VM's shutdown delay in seconds

Signature:

```
void set_shutdown_delay (session ref session_id, VM ref self, int value
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	value	This VM's shutdown delay in seconds

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_snapshot_schedule *Overview:*

Set the value of the snapshot schedule field

Signature:

```
void set_snapshot_schedule (session ref session_id, VM ref self, VMSS
    ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	The VM
VMSS ref	value	The value

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_start_delay *Overview:*

Set this VM's start delay in seconds

Signature:

```
void set_start_delay (session ref session_id, VM ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	The VM
int	value	This VM's start delay in seconds

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_suspend_SR *Overview:*

Set the suspend_SR field of the given VM.

Signature:

```
void set_suspend_SR (session ref session_id, VM ref self, SR ref value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`SR ref`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_suspend_VDI *Overview:*

Set this VM's suspend VDI, which must be indentical to its current one

Signature:

```
void set_suspend_VDI (session ref session_id, VM ref self, VDI ref
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`VDI ref`	value	The suspend VDI uuid

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_tags *Overview:*

Set the tags field of the given VM.

Signature:

```
void set_tags (session ref session_id, VM ref self, string set value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`string set`	value	New value to set

Minimum Role: vm-operator

Return Type: `void`

RPC name: set_user_version *Overview:*

Set the user_version field of the given VM.

Signature:

```
void set_user_version (session ref session_id, VM ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object
`int`	value	New value to set

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_VCPUs_at_startup *Overview:*

Set the number of startup VCPUs for a halted VM

Signature:

```
void set_VCPUs_at_startup (session ref session_id, VM ref self, int
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	value	The new maximum number of VCPUs

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_VCPUs_max *Overview:*

Set the maximum number of VCPUs for a halted VM

Signature:

```
void set_VCPUs_max (session ref session_id, VM ref self, int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM
`int`	value	The new maximum number of VCPUs

Minimum Role: vm-admin

Return Type: `void`

RPC name: set_VCPUs_number_live *Overview:*

Set the number of VCPUs for a running VM

Signature:

```
void set_VCPUs_number_live (session ref session_id, VM ref self, int
    nvcpu)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	The VM
int	nvcpu	The number of VCPUs

Minimum Role: vm-admin

Return Type: **void**

Possible Error Codes: OPERATION_NOT_ALLOWED, LICENCE_RESTRICTION

RPC name: set_VCPUs_params *Overview:*

Set the VCPUs/params field of the given VM.

Signature:

```
void set_VCPUs_params (session ref session_id, VM ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: vm-admin

Return Type: **void**

RPC name: set_xenstore_data *Overview:*

Set the xenstore_data field of the given VM.

Signature:

```
void set_xenstore_data (session ref session_id, VM ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	self	reference to the object
(string -> string)map	value	New value to set

Minimum Role: vm-admin

Return Type: **void**

RPC name: shutdown *Overview:*

Attempts to first clean shutdown a VM and if it should fail then perform a hard shutdown on it.

Signature:

```
void shutdown (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	vm	The VM to shutdown

Minimum Role: client-cert

Return Type: **void**

Possible Error Codes: VM_BAD_POWER_STATE, OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALL
, VM_IS_TEMPLATE

RPC name: snapshot *Overview:*

Snapshots the specified VM, making a new VM. Snapshot automatically exploits the capabilities of the underlying storage repository in which the VM's disk images are stored (e.g. Copy on Write).

Signature:

```
VM ref snapshot (session ref session_id, VM ref vm, string new_name,
    VDI ref set ignore_vdis)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to be snapshotted
`string`	new_name	The name of the snapshotted VM
`VDI ref set`	ignore_vdis	A list of VDIs to ignore for the snapshot

Minimum Role: vm-power-admin

Return Type: `VM ref`

The reference of the newly created VM.

Possible Error Codes: `VM_BAD_POWER_STATE`, `SR_FULL`, `OPERATION_NOT_ALLOWED`

RPC name: snapshot_with_quiesce This message is removed.

Overview:

Snapshots the specified VM with quiesce, making a new VM. Snapshot automatically exploits the capabilities of the underlying storage repository in which the VM's disk images are stored (e.g. Copy on Write).

Signature:

```
VM ref snapshot_with_quiesce (session ref session_id, VM ref vm, string
    new_name)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to be snapshotted
`string`	new_name	The name of the snapshotted VM

Minimum Role: vm-power-admin

Return Type: `VM ref`

The reference of the newly created VM.

Possible Error Codes: `VM_BAD_POWER_STATE`, `SR_FULL`, `OPERATION_NOT_ALLOWED`, `VM_SNAPSHOT_WITH_QUIESCE_FAILED`, `VM_SNAPSHOT_WITH_QUIESCE_TIMEOUT`, `VM_SNAPSHOT_WITH_QUIESCE_PLUGIN_DEOS_NOT_RESPOND`, `VM_SNAPSHOT_WITH_QUIESCE_NOT_`

RPC name: start *Overview:*

Start the specified VM. This function can only be called with the VM is in the Halted State.

Signature:

```
void start (session ref session_id, VM ref vm, bool start_paused, bool
    force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to start
`bool`	start_paused	Instantiate VM in paused state if set to true.
`bool`	force	Attempt to force the VM to start. If this flag is false then the VM may fail pre-boot safety checks (e.g. if the CPU the VM last booted on looks substantially different to the current one)

Minimum Role: vm-operator

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `VM_HVM_REQUIRED`, `VM_IS_TEMPLATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALLOWED`, `BOOTLOADER_FAILED`, `UNKNOWN_BOOTLOADER`, `NO_HOSTS_AVAILABLE`, `LICENCE_RESTRICTION`

RPC name: start_on *Overview:*

Start the specified VM on a particular host. This function can only be called with the VM is in the Halted State.

Signature:

```
void start_on (session ref session_id, VM ref vm, host ref host, bool
    start_paused, bool force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vm	The VM to start
`host ref`	host	The Host on which to start the VM
`bool`	start_paused	Instantiate VM in paused state if set to true.
`bool`	force	Attempt to force the VM to start. If this flag is false then the VM may fail pre-boot safety checks (e.g. if the CPU the VM last booted on looks substantially different to the current one)

Minimum Role: client-cert

Return Type: **`void`**

Possible Error Codes: `VM_BAD_POWER_STATE`, `VM_IS_TEMPLATE`, `OTHER_OPERATION_IN_PROGRESS`, `OPERATION_NOT_ALLOWED`, `BOOTLOADER_FAILED`, `UNKNOWN_BOOTLOADER`

RPC name: suspend *Overview:*

Suspend the specified VM to disk. This can only be called when the specified VM is in the Running state.

Signature:

```
void suspend (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	vm	The VM to suspend

Minimum Role: client-cert

Return Type: **void**

Possible Error Codes: VM_BAD_POWER_STATE, OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALL
, VM_IS_TEMPLATE

RPC name: unpause *Overview:*

Resume the specified VM. This can only be called when the specified VM is in the Paused state.

Signature:

```
void unpause (session ref session_id, VM ref vm)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM ref	vm	The VM to unpause

Minimum Role: vm-operator

Return Type: **void**

Possible Error Codes: VM_BAD_POWER_STATE, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

RPC name: update_allowed_operations *Overview:*

Recomputes the list of acceptable operations

Signature:

```
void update_allowed_operations (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	reference to the object

Minimum Role: pool-admin

Return Type: `void`

RPC name: wait_memory_target_live This message is deprecated.

Overview:

Wait for a running VM to reach its current memory target

Signature:

```
void wait_memory_target_live (session ref session_id, VM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	self	The VM

Minimum Role: read-only

Return Type: `void`

Class: VM_appliance

VM appliance

Fields for class: VM_appliance

Field	Type	Qualifier	Description
allowed_operations	`vm_appliance_operation set`	*RO/runtime*	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.
current_operations	`(string -> vm_appliance_operation )map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VMs	`VM ref set`	*RO/runtime*	all VMs in this appliance

RPCs associated with class: VM_appliance

RPC name: assert_can_be_recovered *Overview:*

Assert whether all SRs required to recover this VM appliance are available.

Signature:

```
void assert_can_be_recovered (session ref session_id, VM_appliance ref
    self, session ref session_to)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance to recover
`session ref`	session_to	The session to which the VM appliance is to be recovered.

Minimum Role: read-only

Return Type: **`void`**

Possible Error Codes: `VM_REQUIRES_SR`

RPC name: clean_shutdown *Overview:*

Perform a clean shutdown of all the VMs in the appliance

Signature:

```
void clean_shutdown (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `OPERATION_PARTIALLY_FAILED`

RPC name: create *Overview:*

Create a new VM_appliance instance, and return its handle.

Signature:

```
VM_appliance ref create (session ref session_id, VM_appliance record
    args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance record`	args	All constructor arguments

Minimum Role: pool-operator

Return Type: `VM_appliance ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified VM_appliance instance.

Signature:

```
void destroy (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the VM_appliances known to the system.

Signature:

```
VM_appliance ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VM_appliance ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VM_appliance references to VM_appliance records for all VM_appliances known to the system.

Signature:

```
(VM_appliance ref -> VM_appliance record) map get_all_records (session
    ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VM_appliance ref -> VM_appliance record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VM_appliance.

Signature:

```
vm_appliance_operation set get_allowed_operations (session ref
    session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vm_appliance_operation set`

value of the field

RPC name: get_by_name_label *Overview:*

Get all the VM_appliance instances with the given label.

Signature:

```
VM_appliance ref set get_by_name_label (session ref session_id, string
    label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
string	label	label of object to return

Minimum Role: read-only

Return Type: `VM_appliance ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the VM_appliance instance with the specified UUID.

Signature:

```
VM_appliance ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
string	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VM_appliance ref`

reference to the object

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VM_appliance.

Signature:

```
(string -> vm_appliance_operation) map get_current_operations (session
    ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vm_appliance_operation)map`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given VM_appliance.

Signature:

```
string get_name_description (session ref session_id, VM_appliance ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given VM_appliance.

Signature:

```
string get_name_label (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VM_appliance.

Signature:

```
VM_appliance record get_record (session ref session_id, VM_appliance
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM_appliance record`

all fields from the object

RPC name: get_SRs_required_for_recovery *Overview:*

Get the list of SRs required by the VM appliance to recover.

Signature:

```
SR ref set get_SRs_required_for_recovery (session ref session_id,
    VM_appliance ref self, session ref session_to)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance for which the required list of SRs has to be recovered.
`session ref`	session_to	The session to which the list of SRs have to be recovered .

Minimum Role: read-only

Return Type: `SR ref set`

refs for SRs required to recover the VM

RPC name: get_uuid *Overview:*

Get the uuid field of the given VM_appliance.

Signature:

```
string get_uuid (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VMs *Overview:*

Get the VMs field of the given VM_appliance.

Signature:

```
VM ref set get_VMs (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref set`

value of the field

RPC name: hard_shutdown *Overview:*

Perform a hard shutdown of all the VMs in the appliance

Signature:

```
void hard_shutdown (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `OPERATION_PARTIALLY_FAILED`

RPC name: recover *Overview:*

Recover the VM appliance

Signature:

```
void recover (session ref session_id, VM_appliance ref self, session
    ref session_to, bool force)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance to recover
`session ref`	session_to	The session to which the VM appliance is to be recovered.
`bool`	force	Whether the VMs should replace newer versions of themselves.

Minimum Role: read-only

Return Type: **`void`**

Possible Error Codes: `VM_REQUIRES_SR`

RPC name: set_name_description *Overview:*

Set the name/description field of the given VM_appliance.

Signature:

```
void set_name_description (session ref session_id, VM_appliance ref
    self, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_name_label *Overview:*

Set the name/label field of the given VM_appliance.

Signature:

```
void set_name_label (session ref session_id, VM_appliance ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: shutdown *Overview:*

For each VM in the appliance, try to shut it down cleanly. If this fails, perform a hard shutdown of the VM.

Signature:

```
void shutdown (session ref session_id, VM_appliance ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance

Minimum Role: pool-operator

Return Type: `void`

Possible Error Codes: `OPERATION_PARTIALLY_FAILED`

RPC name: start *Overview:*

Start all VMs in the appliance

Signature:

```
void start (session ref session_id, VM_appliance ref self, bool paused)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_appliance ref`	self	The VM appliance
`bool`	paused	Instantiate all VMs belonging to this appliance in paused state if set to true.

Minimum Role: pool-operator

Return Type: **`void`**

Possible Error Codes: `OPERATION_PARTIALLY_FAILED`

Class: VM_guest_metrics

The metrics reported by the guest (as opposed to inferred from outside)

Fields for class: VM_guest_metrics

Field	Type	Qualifier	Description
can_use_hotplug_vbd	`tristate_type`	*RO/runtime*	The guest's statement of whether it supports VBD hotplug, i.e. whether it is capable of responding immediately to instantiation of a new VBD by bringing online a new PV block device. If the guest states that it is not capable, then the VBD plug and unplug operations will not be allowed while the guest is running.
can_use_hotplug_vif	`tristate_type`	*RO/runtime*	The guest's statement of whether it supports VIF hotplug, i.e. whether it is capable of responding immediately to instantiation of a new VIF by bringing online a new PV network device. If the guest states that it is not capable, then the VIF plug and unplug operations will not be allowed while the guest is running.
disks	`(string -> string)map`	*RO/runtime*	**Removed**. This field exists but has no data.
last_updated	`datetime`	*RO/runtime*	Time at which this information was last updated

Field	Type	Qualifier	Description
live	`bool`	*RO/runtime*	True if the guest is sending heartbeat messages via the guest agent
memory	`(string -> string)map`	*RO/runtime*	**Removed**. This field exists but has no data. Use the memory and memory_internal_free RRD data-sources instead.
networks	`(string -> string)map`	*RO/runtime*	network configuration
os_version	`(string -> string)map`	*RO/runtime*	version of the OS
other	`(string -> string)map`	*RO/runtime*	anything else
other_config	`(string -> string)map`	*RW*	additional configuration
PV_drivers_detected	`bool`	*RO/runtime*	At least one of the guest's devices has successfully connected to the backend.
PV_drivers_up_to_date	`bool`	*RO/runtime*	**Deprecated**. Logically equivalent to PV_drivers_detected
PV_drivers_version	`(string -> string)map`	*RO/runtime*	version of the PV drivers
uuid	`string`	*RO/runtime*	Unique identifier/object reference

RPCs associated with class: VM_guest_metrics

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VM_guest_metrics.

Signature:

```
void add_to_other_config (session ref session_id, VM_guest_metrics ref
    self, string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the VM_guest_metrics instances known to the system.

Signature:

```
VM_guest_metrics ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VM_guest_metrics ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VM_guest_metrics references to VM_guest_metrics records for all VM_guest_metrics instances known to the system.

Signature:

```
(VM_guest_metrics ref -> VM_guest_metrics record) map get_all_records (
    session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VM_guest_metrics ref -> VM_guest_metrics record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the VM_guest_metrics instance with the specified UUID.

Signature:

```
VM_guest_metrics ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VM_guest_metrics ref`

reference to the object

RPC name: get_can_use_hotplug_vbd *Overview:*

Get the can_use_hotplug_vbd field of the given VM_guest_metrics.

Signature:

```
tristate_type get_can_use_hotplug_vbd (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `tristate_type`

value of the field

RPC name: get_can_use_hotplug_vif *Overview:*

Get the can_use_hotplug_vif field of the given VM_guest_metrics.

Signature:

```
tristate_type get_can_use_hotplug_vif (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `tristate_type`

value of the field

RPC name: get_disks **This message is removed.**

Overview:

Get the disks field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_disks (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_last_updated *Overview:*

Get the last_updated field of the given VM_guest_metrics.

Signature:

```
datetime get_last_updated (session ref session_id, VM_guest_metrics ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_live *Overview:*

Get the live field of the given VM_guest_metrics.

Signature:

```
bool get_live (session ref session_id, VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_memory **This message is removed.**

Overview:

Get the memory field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_memory (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_networks *Overview:*

Get the networks field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_networks (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_os_version *Overview:*

Get the os_version field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_os_version (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_other *Overview:*

Get the other field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_other (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_PV_drivers_detected *Overview:*

Get the PV_drivers_detected field of the given VM_guest_metrics.

Signature:

```
bool get_PV_drivers_detected (session ref session_id, VM_guest_metrics
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_PV_drivers_up_to_date **This message is deprecated.**

Overview:

Get the PV_drivers_up_to_date field of the given VM_guest_metrics.

Signature:

```
bool get_PV_drivers_up_to_date (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_PV_drivers_version *Overview:*

Get the PV_drivers_version field of the given VM_guest_metrics.

Signature:

```
(string -> string) map get_PV_drivers_version (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VM_guest_metrics.

Signature:

```
VM_guest_metrics record get_record (session ref session_id,
    VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM_guest_metrics record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given VM_guest_metrics.

Signature:

```
string get_uuid (session ref session_id, VM_guest_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM_guest_metrics ref	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VM_guest_metrics. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VM_guest_metrics
     ref self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM_guest_metrics ref	self	reference to the object
string	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VM_guest_metrics.

Signature:

```
void set_other_config (session ref session_id, VM_guest_metrics ref
    self, (string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_guest_metrics ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

Class: VM_metrics

The metrics associated with a VM

Fields for class: VM_metrics

Field	Type	Qualifier	Description
current_domain_type	`domain_type`	*RO/runtime*	The current domain type of the VM (for running,suspended, or paused VMs). The last-known domain type for halted VMs.
hvm	`bool`	*RO/runtime*	hardware virtual machine
install_time	`datetime`	*RO/runtime*	Time at which the VM was installed
last_updated	`datetime`	*RO/runtime*	Time at which this information was last updated
memory_actual	**`int`**	*RO/runtime*	Guest's actual memory (bytes)
nested_virt	`bool`	*RO/runtime*	VM supports nested virtualisation

Field	Type	Qualifier	Description
nomigrate	bool	*RO/runtime*	VM is immobile and can't migrate between hosts
other_config	(string -> string)map	*RW*	additional configuration
start_time	datetime	*RO/runtime*	Time at which this VM was last booted
state	string set	*RO/runtime*	The state of the guest, eg blocked, dying etc
uuid	string	*RO/runtime*	Unique identifier/object reference
VCPUs_CPU	(int -> int)map	*RO/runtime*	VCPU to PCPU map
VCPUs_flags	(int -> string set)map	*RO/runtime*	CPU flags (blocked,online,running)
VCPUs_number	int	*RO/runtime*	Current number of VCPUs
VCPUs_params	(string -> string)map	*RO/runtime*	The live equivalent to VM.VCPUs_params
VCPUs_utilisation	(int -> float) map	*RO/runtime*	**Removed**. Utilisation for all of guest's current VCPUs

RPCs associated with class: VM_metrics

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VM_metrics.

Signature:

```
void add_to_other_config (session ref session_id, VM_metrics ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the VM_metrics instances known to the system.

Signature:

```
VM_metrics ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VM_metrics ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VM_metrics references to VM_metrics records for all VM_metrics instances known to the system.

Signature:

```
(VM_metrics ref -> VM_metrics record) map get_all_records (session ref
    session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VM_metrics ref -> VM_metrics record)map`

records of all objects

RPC name: get_by_uuid *Overview:*

Get a reference to the VM_metrics instance with the specified UUID.

Signature:

```
VM_metrics ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VM_metrics ref`

reference to the object

RPC name: get_current_domain_type *Overview:*

Get the current_domain_type field of the given VM_metrics.

Signature:

```
domain_type get_current_domain_type (session ref session_id, VM_metrics
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `domain_type`

value of the field

RPC name: get_hvm *Overview:*

Get the hvm field of the given VM_metrics.

Signature:

```
bool get_hvm (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_install_time *Overview:*

Get the install_time field of the given VM_metrics.

Signature:

```
datetime get_install_time (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_last_updated *Overview:*

Get the last_updated field of the given VM_metrics.

Signature:

```
datetime get_last_updated (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_memory_actual *Overview:*

Get the memory/actual field of the given VM_metrics.

Signature:

```
int get_memory_actual (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_nested_virt *Overview:*

Get the nested_virt field of the given VM_metrics.

Signature:

```
bool get_nested_virt (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_nomigrate *Overview:*

Get the nomigrate field of the given VM_metrics.

Signature:

```
bool get_nomigrate (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VM_metrics.

Signature:

```
(string -> string) map get_other_config (session ref session_id,
    VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VM_metrics.

Signature:

```
VM_metrics record get_record (session ref session_id, VM_metrics ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM_metrics record`

all fields from the object

RPC name: get_start_time *Overview:*

Get the start_time field of the given VM_metrics.

Signature:

```
datetime get_start_time (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_state *Overview:*

Get the state field of the given VM_metrics.

Signature:

```
string set get_state (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VM_metrics.

Signature:

```
string get_uuid (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VCPUs_CPU *Overview:*

Get the VCPUs/CPU field of the given VM_metrics.

Signature:

```
(int -> int) map get_VCPUs_CPU (session ref session_id, VM_metrics ref
   self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(int -> int)map`

value of the field

RPC name: get_VCPUs_flags *Overview:*

Get the VCPUs/flags field of the given VM_metrics.

Signature:

```
(int -> string set) map get_VCPUs_flags (session ref session_id,
    VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(int -> string set)map`

value of the field

RPC name: get_VCPUs_number *Overview:*

Get the VCPUs/number field of the given VM_metrics.

Signature:

```
int get_VCPUs_number (session ref session_id, VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_VCPUs_params *Overview:*

Get the VCPUs/params field of the given VM_metrics.

Signature:

```
(string -> string) map get_VCPUs_params (session ref session_id,
    VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM_metrics ref	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_VCPUs_utilisation **This message is removed.**

Overview:

Get the VCPUs/utilisation field of the given VM_metrics.

Signature:

```
(int -> float) map get_VCPUs_utilisation (session ref session_id,
    VM_metrics ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VM_metrics ref	self	reference to the object

Minimum Role: read-only

Return Type: `(int -> float)map`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VM_metrics. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VM_metrics ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object
`string`	key	Key to remove

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VM_metrics.

Signature:

```
void set_other_config (session ref session_id, VM_metrics ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM_metrics ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: vm-admin

Return Type: **`void`**

Class: VMPP

This class is removed.

VM Protection Policy

Fields for class: VMPP

Field	Type	Qualifier	Description
alarm_config	`(string -> string) map`	*RO/constructor*	**Removed**. configuration for the alarm
archive_frequency	`vmpp_archive_frequency`	*RO/constructor*	**Removed**. frequency of the archive schedule
archive_last_run_time	`datetime`	*RO/runtime*	**Removed**. time of the last archive
archive_schedule	`(string -> string) map`	*RO/constructor*	**Removed**. schedule of the archive containing 'hour', 'min', 'days'. Date/time-related information is in Local Timezone
archive_target_config	`(string -> string) map`	*RO/constructor*	**Removed**. configuration for the archive, including its 'location', 'username', 'password'
archive_target_type	`vmpp_archive_target_type`	*RO/constructor*	**Removed**. type of the archive target config
backup_frequency	`vmpp_backup_frequency`	*RO/constructor*	**Removed**. frequency of the backup schedule
backup_last_run_time	`datetime`	*RO/runtime*	**Removed**. time of the last backup

Field	Type	Qualifier	Description
backup_retention_value	`int`	*RO/constructor*	**Removed**. maximum number of backups that should be stored at any time
backup_schedule	`(string -> string) map`	*RO/constructor*	**Removed**. schedule of the backup containing 'hour', 'min', 'days'. Date/time-related information is in Local Timezone
backup_type	`vmpp_backup_type`	*RW*	**Removed**. type of the backup sub-policy
is_alarm_enabled	`bool`	*RO/constructor*	**Removed**. true if alarm is enabled for this policy
is_archive_running	`bool`	*RO/runtime*	**Removed**. true if this protection policy's archive is running
is_backup_running	`bool`	*RO/runtime*	**Removed**. true if this protection policy's backup is running
is_policy_enabled	`bool`	*RW*	**Removed**. enable or disable this policy
name_description	`string`	*RW*	**Removed**. a notes field containing human-readable description
name_label	`string`	*RW*	**Removed**. a human-readable name
recent_alerts	`string set`	*RO/runtime*	**Removed**. recent alerts
uuid	`string`	*RO/runtime*	**Removed**. Unique identifier/object reference

Field	Type	Qualifier	Description
VMs	`VM ref set`	*RO/runtime*	**Removed**. all VMs attached to this protection policy

RPCs associated with class: VMPP

RPC name: add_to_alarm_config   This message is removed.

Overview:

Signature:

```
void add_to_alarm_config (session ref session_id, VMPP ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`string`	key	the key to add
`string`	value	the value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: add_to_archive_schedule   This message is removed.

Overview:

Signature:

```
void add_to_archive_schedule (session ref session_id, VMPP ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMPP ref	self	The protection policy
string	key	the key to add
string	value	the value to add

Minimum Role: pool-operator

Return Type: **void**

RPC name: add_to_archive_target_config This message is removed.

Overview:

Signature:

```
void add_to_archive_target_config (session ref session_id, VMPP ref
    self, string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMPP ref	self	The protection policy
string	key	the key to add
string	value	the value to add

Minimum Role: pool-operator

Return Type: **void**

RPC name: add_to_backup_schedule This message is removed.

Overview:

Signature:

```
void add_to_backup_schedule (session ref session_id, VMPP ref self,
    string key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`string`	key	the key to add
`string`	value	the value to add

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: archive_now This message is removed.

Overview:

This call archives the snapshot provided as a parameter

Signature:

```
string archive_now (session ref session_id, VM ref snapshot)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	snapshot	The snapshot to archive

Minimum Role: vm-power-admin

Return Type: `string`

An XMLRPC result

RPC name: create This message is removed.

Overview:

Create a new VMPP instance, and return its handle.

Signature:

```
VMPP ref create (session ref session_id, VMPP record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP record`	args	All constructor arguments

Minimum Role: pool-operator

Return Type: `VMPP ref`

reference to the newly created object

RPC name: destroy This message is removed.

Overview:

Destroy the specified VMPP instance.

Signature:

```
void destroy (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_alarm_config This message is removed.

Overview:

Get the alarm_config field of the given VMPP.

Signature:

```
(string -> string) map get_alarm_config (session ref session_id, VMPP
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_alerts This message is removed.

Overview:

This call fetches a history of alerts for a given protection policy

Signature:

```
string set get_alerts (session ref session_id, VMPP ref vmpp, int
    hours_from_now)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	vmpp	The protection policy
`int`	hours_from_now	how many hours in the past the oldest record to fetch is

Minimum Role: pool-operator

Return Type: `string set`

A list of alerts encoded in xml

RPC name: get_all   This message is removed.

Overview:

Return a list of all the VMPPs known to the system.

Signature:

```
VMPP ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VMPP ref set`

references to all objects

RPC name: get_all_records   This message is removed.

Overview:

Return a map of VMPP references to VMPP records for all VMPPs known to the system.

Signature:

```
(VMPP ref -> VMPP record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VMPP ref -> VMPP record)map`

records of all objects

RPC name: get_archive_frequency   This message is removed.

Overview:

Get the archive_frequency field of the given VMPP.

Signature:

```
vmpp_archive_frequency get_archive_frequency (session ref session_id,
    VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vmpp_archive_frequency`

value of the field

RPC name: get_archive_last_run_time This message is removed.

Overview:

Get the archive_last_run_time field of the given VMPP.

Signature:

```
datetime get_archive_last_run_time (session ref session_id, VMPP ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_archive_schedule This message is removed.

Overview:

Get the archive_schedule field of the given VMPP.

Signature:

```
(string -> string) map get_archive_schedule (session ref session_id,
    VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMPP ref	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_archive_target_config This message is removed.

Overview:

Get the archive_target_config field of the given VMPP.

Signature:

```
(string -> string) map get_archive_target_config (session ref
    session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMPP ref	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_archive_target_type This message is removed.

Overview:

Get the archive_target_type field of the given VMPP.

Signature:

```
vmpp_archive_target_type get_archive_target_type (session ref
    session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vmpp_archive_target_type`

value of the field

RPC name: get_backup_frequency **This message is removed.**

Overview:

Get the backup_frequency field of the given VMPP.

Signature:

```
vmpp_backup_frequency get_backup_frequency (session ref session_id,
    VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vmpp_backup_frequency`

value of the field

RPC name: get_backup_last_run_time This message is removed.

Overview:

Get the backup_last_run_time field of the given VMPP.

Signature:

```
datetime get_backup_last_run_time (session ref session_id, VMPP ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_backup_retention_value This message is removed.

Overview:

Get the backup_retention_value field of the given VMPP.

Signature:

```
int get_backup_retention_value (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_backup_schedule **This message is removed.**

Overview:

Get the backup_schedule field of the given VMPP.

Signature:

```
(string -> string) map get_backup_schedule (session ref session_id,
    VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_backup_type **This message is removed.**

Overview:

Get the backup_type field of the given VMPP.

Signature:

```
vmpp_backup_type get_backup_type (session ref session_id, VMPP ref self
    )
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vmpp_backup_type`

value of the field

RPC name: get_by_name_label This message is removed.

Overview:

Get all the VMPP instances with the given label.

Signature:

```
VMPP ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `VMPP ref set`

references to objects with matching names

RPC name: get_by_uuid This message is removed.

Overview:

Get a reference to the VMPP instance with the specified UUID.

Signature:

```
VMPP ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VMPP ref`

reference to the object

RPC name: get_is_alarm_enabled This message is removed.

Overview:

Get the is_alarm_enabled field of the given VMPP.

Signature:

```
bool get_is_alarm_enabled (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMPP ref	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_archive_running This message is removed.

Overview:

Get the is_archive_running field of the given VMPP.

Signature:

```
bool get_is_archive_running (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMPP ref	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_backup_running This message is removed.

Overview:

Get the is_backup_running field of the given VMPP.

Signature:

```
bool get_is_backup_running (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_policy_enabled This message is removed.

Overview:

Get the is_policy_enabled field of the given VMPP.

Signature:

```
bool get_is_policy_enabled (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_name_description   This message is removed.

Overview:

Get the name/description field of the given VMPP.

Signature:

```
string get_name_description (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label   This message is removed.

Overview:

Get the name/label field of the given VMPP.

Signature:

```
string get_name_label (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_recent_alerts This message is removed.

Overview:

Get the recent_alerts field of the given VMPP.

Signature:

```
string set get_recent_alerts (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string set`

value of the field

RPC name: get_record This message is removed.

Overview:

Get a record containing the current state of the given VMPP.

Signature:

```
VMPP record get_record (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VMPP record`

all fields from the object

RPC name: get_uuid This message is removed.

Overview:

Get the uuid field of the given VMPP.

Signature:

```
string get_uuid (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VMs This message is removed.

Overview:

Get the VMs field of the given VMPP.

Signature:

```
VM ref set get_VMs (session ref session_id, VMPP ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref set`

value of the field

RPC name: protect_now This message is removed.

Overview:

This call executes the protection policy immediately

Signature:

```
string protect_now (session ref session_id, VMPP ref vmpp)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	vmpp	The protection policy to execute

Minimum Role: pool-operator

Return Type: `string`

An XMLRPC result

RPC name: remove_from_alarm_config This message is removed.

Overview:

Signature:

```
void remove_from_alarm_config (session ref session_id, VMPP ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`string`	key	the key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: remove_from_archive_schedule   This message is removed.

Overview:

Signature:

```
void remove_from_archive_schedule (session ref session_id, VMPP ref
    self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`string`	key	the key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_from_archive_target_config   This message is removed.

Overview:

Signature:

```
void remove_from_archive_target_config (session ref session_id, VMPP
    ref self, string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`string`	key	the key to remove

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: remove_from_backup_schedule **This message is removed.**

Overview:

Signature:

```
void remove_from_backup_schedule (session ref session_id, VMPP ref self
    , string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`string`	key	the key to remove

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_alarm_config **This message is removed.**

Overview:

Signature:

```
void set_alarm_config (session ref session_id, VMPP ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`(string -> string)map`	value	the value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_archive_frequency This message is removed.

Overview:

Set the value of the archive_frequency field

Signature:

```
void set_archive_frequency (session ref session_id, VMPP ref self,
    vmpp_archive_frequency value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`vmpp_archive_frequency`	value	the archive frequency

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_archive_last_run_time This message is removed.

Overview:

Signature:

```
void set_archive_last_run_time (session ref session_id, VMPP ref self,
    datetime value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`datetime`	value	the value to set

Return Type: `void`

RPC name: set_archive_schedule This message is removed.

Overview:

Signature:

```
void set_archive_schedule (session ref session_id, VMPP ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`(string -> string)map`	value	the value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_archive_target_config This message is removed.

Overview:

Signature:

```
void set_archive_target_config (session ref session_id, VMPP ref self,
    (string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`(string -> string)map`	value	the value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_archive_target_type This message is removed.

Overview:

Set the value of the archive_target_config_type field

Signature:

```
void set_archive_target_type (session ref session_id, VMPP ref self,
    vmpp_archive_target_type value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`vmpp_archive_target_type`	value	the archive target config type

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_backup_frequency This message is removed.

Overview:

Set the value of the backup_frequency field

Signature:

```
void set_backup_frequency (session ref session_id, VMPP ref self,
    vmpp_backup_frequency value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`vmpp_backup_frequency`	value	the backup frequency

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_backup_last_run_time   This message is removed.

Overview:

Signature:

```
void set_backup_last_run_time (session ref session_id, VMPP ref self,
    datetime value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`datetime`	value	the value to set

Return Type: `void`

RPC name: set_backup_retention_value   This message is removed.

Overview:

Signature:

```
void set_backup_retention_value (session ref session_id, VMPP ref self,
     int value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`int`	value	the value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_backup_schedule **This message is removed.**

Overview:

Signature:

```
void set_backup_schedule (session ref session_id, VMPP ref self, (
    string -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`(string -> string)map`	value	the value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_backup_type **This message is removed.**

Overview:

Set the backup_type field of the given VMPP.

Signature:

```
void set_backup_type (session ref session_id, VMPP ref self,
    vmpp_backup_type value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object
`vmpp_backup_type`	value	New value to set

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_is_alarm_enabled **This message is removed.**

Overview:

Set the value of the is_alarm_enabled field

Signature:

```
void set_is_alarm_enabled (session ref session_id, VMPP ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	The protection policy
`bool`	value	true if alarm is enabled for this policy

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_is_policy_enabled **This message is removed.**

Overview:

Set the is_policy_enabled field of the given VMPP.

Signature:

```
void set_is_policy_enabled (session ref session_id, VMPP ref self, bool
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object
`bool`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_name_description This message is removed.

Overview:

Set the name/description field of the given VMPP.

Signature:

```
void set_name_description (session ref session_id, VMPP ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object
`string`	value	New value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_name_label This message is removed.

Overview:

Set the name/label field of the given VMPP.

Signature:

```
void set_name_label (session ref session_id, VMPP ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMPP ref`	self	reference to the object
`string`	value	New value to set

type	name	description

Minimum Role: pool-operator

Return Type: `void`

Class: VMSS

VM Snapshot Schedule

Fields for class: VMSS

Field	Type	Qualifier	Description
enabled	`bool`	*RW*	enable or disable this snapshot schedule
frequency	`vmss_frequency`	*RO/constructor*	frequency of taking snapshot from snapshot schedule
last_run_time	`datetime`	*RO/runtime*	time of the last snapshot
name_description	`string`	*RW*	a notes field containing human-readable description
name_label	`string`	*RW*	a human-readable name
retained_snapshots	`int`	*RO/constructor*	maximum number of snapshots that should be stored at any time
schedule	`(string -> string) map`	*RO/constructor*	schedule of the snapshot containing 'hour', 'min', 'days'. Date/time-related information is in Local Timezone
type	`vmss_type`	*RO/constructor*	type of the snapshot schedule

Field	Type	Qualifier	Description
uuid	string	*RO/runtime*	Unique identifier/object reference
VMs	VM ref set	*RO/runtime*	all VMs attached to this snapshot schedule

RPCs associated with class: VMSS

RPC name: add_to_schedule *Overview:*

Signature:

```
void add_to_schedule (session ref session_id, VMSS ref self, string key
    , string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMSS ref	self	The snapshot schedule
string	key	the key to add
string	value	the value to add

Minimum Role: pool-operator

Return Type: **void**

RPC name: create *Overview:*

Create a new VMSS instance, and return its handle.

Signature:

```
VMSS ref create (session ref session_id, VMSS record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS record`	args	All constructor arguments

Minimum Role: pool-operator

Return Type: `VMSS ref`

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified VMSS instance.

Signature:

```
void destroy (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the VMSSs known to the system.

Signature:

```
VMSS ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VMSS ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VMSS references to VMSS records for all VMSSs known to the system.

Signature:

```
(VMSS ref -> VMSS record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VMSS ref -> VMSS record)map`

records of all objects

RPC name: get_by_name_label *Overview:*

Get all the VMSS instances with the given label.

Signature:

```
VMSS ref set get_by_name_label (session ref session_id, string label)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	label	label of object to return

Minimum Role: read-only

Return Type: `VMSS ref set`

references to objects with matching names

RPC name: get_by_uuid *Overview:*

Get a reference to the VMSS instance with the specified UUID.

Signature:

```
VMSS ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VMSS ref`

reference to the object

RPC name: get_enabled *Overview:*

Get the enabled field of the given VMSS.

Signature:

```
bool get_enabled (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_frequency *Overview:*

Get the frequency field of the given VMSS.

Signature:

```
vmss_frequency get_frequency (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vmss_frequency`

value of the field

RPC name: get_last_run_time *Overview:*

Get the last_run_time field of the given VMSS.

Signature:

```
datetime get_last_run_time (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `datetime`

value of the field

RPC name: get_name_description *Overview:*

Get the name/description field of the given VMSS.

Signature:

```
string get_name_description (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_name_label *Overview:*

Get the name/label field of the given VMSS.

Signature:

```
string get_name_label (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VMSS.

Signature:

```
VMSS record get_record (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VMSS record`

all fields from the object

RPC name: get_retained_snapshots *Overview:*

Get the retained_snapshots field of the given VMSS.

Signature:

```
int get_retained_snapshots (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `int`

value of the field

RPC name: get_schedule *Overview:*

Get the schedule field of the given VMSS.

Signature:

```
(string -> string) map get_schedule (session ref session_id, VMSS ref
    self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_type *Overview:*

Get the type field of the given VMSS.

Signature:

```
vmss_type get_type (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vmss_type`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VMSS.

Signature:

```
string get_uuid (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VMs *Overview:*

Get the VMs field of the given VMSS.

Signature:

```
VM ref set get_VMs (session ref session_id, VMSS ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref set`

value of the field

RPC name: remove_from_schedule *Overview:*

Signature:

```
void remove_from_schedule (session ref session_id, VMSS ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMSS ref	self	The snapshot schedule
string	key	the key to remove

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_enabled *Overview:*

Set the enabled field of the given VMSS.

Signature:

```
void set_enabled (session ref session_id, VMSS ref self, bool value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMSS ref	self	reference to the object
bool	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_frequency *Overview:*

Set the value of the frequency field

Signature:

```
void set_frequency (session ref session_id, VMSS ref self,
    vmss_frequency value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	The snapshot schedule
`vmss_frequency`	value	the snapshot schedule frequency

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: set_last_run_time *Overview:*

Signature:

```
void set_last_run_time (session ref session_id, VMSS ref self, datetime
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	The snapshot schedule
`datetime`	value	the value to set

Return Type: **`void`**

RPC name: set_name_description *Overview:*

Set the name/description field of the given VMSS.

Signature:

```
void set_name_description (session ref session_id, VMSS ref self,
    string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMSS ref	self	reference to the object
string	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_name_label *Overview:*

Set the name/label field of the given VMSS.

Signature:

```
void set_name_label (session ref session_id, VMSS ref self, string
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VMSS ref	self	reference to the object
string	value	New value to set

Minimum Role: pool-operator

Return Type: **void**

RPC name: set_retained_snapshots *Overview:*

Signature:

```
void set_retained_snapshots (session ref session_id, VMSS ref self, int
    value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	The schedule snapshot
`int`	value	the value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_schedule *Overview:*

Signature:

```
void set_schedule (session ref session_id, VMSS ref self, (string ->
    string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	The snapshot schedule
`(string -> string)map`	value	the value to set

Minimum Role: pool-operator

Return Type: `void`

RPC name: set_type *Overview:*

Signature:

```
void set_type (session ref session_id, VMSS ref self, vmss_type value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	self	The snapshot schedule
`vmss_type`	value	the snapshot schedule type

Minimum Role: pool-operator

Return Type: **`void`**

RPC name: snapshot_now *Overview:*

This call executes the snapshot schedule immediately

Signature:

```
string snapshot_now (session ref session_id, VMSS ref vmss)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VMSS ref`	vmss	Snapshot Schedule to execute

Minimum Role: pool-operator

Return Type: `string`

An XMLRPC result

Class: VTPM

A virtual TPM device

Fields for class: VTPM

Field	Type	Qualifier	Description
allowed_operations	`vtpm_operations set`	*RO/runtime*	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.
backend	`VM ref`	*RO/runtime*	The domain where the backend is located (unused)
current_operations	`(string -> vtpm_operations ) map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
is_protected	`bool`	*RO/runtime*	Whether the contents of the VTPM are secured according to the TPM spec
is_unique	`bool`	*RO/constructor*	Whether the contents are never copied, satisfying the TPM spec
persistence_backend	`persistence_backend`	*RO/runtime*	The backend where the vTPM is persisted
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VM	`VM ref`	*RO/constructor*	The virtual machine the TPM is attached to

RPCs associated with class: VTPM

RPC name: create *Overview:*

Create a new VTPM instance, and return its handle.

Signature:

```
VTPM ref create (session ref session_id, VM ref vM, bool is_unique)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	vM	The VM reference the VTPM will be attached to
`bool`	is_unique	Whether the VTPM must be unique

Minimum Role: vm-admin

Return Type: `VTPM ref`

The reference of the newly created VTPM

RPC name: destroy *Overview:*

Destroy the specified VTPM instance, along with its state.

Signature:

```
void destroy (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	The reference to the VTPM object

Minimum Role: vm-admin

Return Type: **`void`**

RPC name: get_all *Overview:*

Return a list of all the VTPMs known to the system.

Signature:

```
VTPM ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VTPM ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VTPM references to VTPM records for all VTPMs known to the system.

Signature:

```
(VTPM ref -> VTPM record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VTPM ref -> VTPM record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VTPM.

Signature:

```
vtpm_operations set get_allowed_operations (session ref session_id,
    VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vtpm_operations set`

value of the field

RPC name: get_backend *Overview:*

Get the backend field of the given VTPM.

Signature:

```
VM ref get_backend (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the VTPM instance with the specified UUID.

Signature:

```
VTPM ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VTPM ref`

reference to the object

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VTPM.

Signature:

```
(string -> vtpm_operations) map get_current_operations (session ref
    session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vtpm_operations)map`

value of the field

RPC name: get_is_protected *Overview:*

Get the is_protected field of the given VTPM.

Signature:

```
bool get_is_protected (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_is_unique *Overview:*

Get the is_unique field of the given VTPM.

Signature:

```
bool get_is_unique (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_persistence_backend *Overview:*

Get the persistence_backend field of the given VTPM.

Signature:

```
persistence_backend get_persistence_backend (session ref session_id,
    VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `persistence_backend`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VTPM.

Signature:

```
VTPM record get_record (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VTPM record`

all fields from the object

RPC name: get_uuid *Overview:*

Get the uuid field of the given VTPM.

Signature:

```
string get_uuid (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VM *Overview:*

Get the VM field of the given VTPM.

Signature:

```
VM ref get_VM (session ref session_id, VTPM ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VTPM ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

Class: VUSB

Describes the vusb device

Fields for class: VUSB

Field	Type	Qualifier	Description
allowed_operations	`vusb_operations set`	*RO/runtime*	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.

Field	Type	Qualifier	Description
current_operations	`(string -> vusb_operations)map`	*RO/runtime*	links each of the running tasks using this object (by reference) to a current_operation enum which describes the nature of the task.
currently_attached	`bool`	*RO/runtime*	is the device currently attached
other_config	`(string -> string)map`	*RW*	Additional configuration
USB_group	`USB_group ref`	*RO/runtime*	USB group used by the VUSB
uuid	`string`	*RO/runtime*	Unique identifier/object reference
VM	`VM ref`	*RO/runtime*	VM that owns the VUSB

RPCs associated with class: VUSB

RPC name: add_to_other_config *Overview:*

Add the given key-value pair to the other_config field of the given VUSB.

Signature:

```
void add_to_other_config (session ref session_id, VUSB ref self, string
    key, string value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object
`string`	key	Key to add
`string`	value	Value to add

Minimum Role: pool-admin

Return Type: `void`

RPC name: create *Overview:*

Create a new VUSB record in the database only

Signature:

```
VUSB ref create (session ref session_id, VM ref VM, USB_group ref
    USB_group, (string -> string) map other_config)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VM ref`	VM	The VM
`USB_group ref`	USB_group	
`(string -> string)map`	other_config	

Minimum Role: pool-admin

Return Type: `VUSB ref`

The ref of the newly created VUSB record.

RPC name: destroy *Overview:*

Removes a VUSB record from the database

Signature:

```
void destroy (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	The VUSB to destroy about

Minimum Role: pool-admin

Return Type: `void`

RPC name: get_all *Overview:*

Return a list of all the VUSBs known to the system.

Signature:

```
VUSB ref set get_all (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `VUSB ref set`

references to all objects

RPC name: get_all_records *Overview:*

Return a map of VUSB references to VUSB records for all VUSBs known to the system.

Signature:

```
(VUSB ref -> VUSB record) map get_all_records (session ref session_id)
<!--NeedCopy-->
```

Minimum Role: read-only

Return Type: `(VUSB ref -> VUSB record)map`

records of all objects

RPC name: get_allowed_operations *Overview:*

Get the allowed_operations field of the given VUSB.

Signature:

```
vusb_operations set get_allowed_operations (session ref session_id,
    VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session

type	name	description
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `vusb_operations set`

value of the field

RPC name: get_by_uuid *Overview:*

Get a reference to the VUSB instance with the specified UUID.

Signature:

```
VUSB ref get_by_uuid (session ref session_id, string uuid)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`string`	uuid	UUID of object to return

Minimum Role: read-only

Return Type: `VUSB ref`

reference to the object

RPC name: get_current_operations *Overview:*

Get the current_operations field of the given VUSB.

Signature:

```
(string -> vusb_operations) map get_current_operations (session ref
    session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> vusb_operations)map`

value of the field

RPC name: get_currently_attached *Overview:*

Get the currently_attached field of the given VUSB.

Signature:

```
bool get_currently_attached (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `bool`

value of the field

RPC name: get_other_config *Overview:*

Get the other_config field of the given VUSB.

Signature:

```
(string -> string) map get_other_config (session ref session_id, VUSB
    ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `(string -> string)map`

value of the field

RPC name: get_record *Overview:*

Get a record containing the current state of the given VUSB.

Signature:

```
VUSB record get_record (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `VUSB record`

all fields from the object

RPC name: get_USB_group *Overview:*

Get the USB_group field of the given VUSB.

Signature:

```
USB_group ref get_USB_group (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `USB_group ref`

value of the field

RPC name: get_uuid *Overview:*

Get the uuid field of the given VUSB.

Signature:

```
string get_uuid (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object

Minimum Role: read-only

Return Type: `string`

value of the field

RPC name: get_VM *Overview:*

Get the VM field of the given VUSB.

Signature:

```
VM ref get_VM (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VUSB ref	self	reference to the object

Minimum Role: read-only

Return Type: `VM ref`

value of the field

RPC name: remove_from_other_config *Overview:*

Remove the given key and its corresponding value from the other_config field of the given VUSB. If the key is not in that Map, then do nothing.

Signature:

```
void remove_from_other_config (session ref session_id, VUSB ref self,
    string key)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VUSB ref	self	reference to the object
string	key	Key to remove

Minimum Role: pool-admin

Return Type: **`void`**

RPC name: set_other_config *Overview:*

Set the other_config field of the given VUSB.

Signature:

```
void set_other_config (session ref session_id, VUSB ref self, (string
    -> string) map value)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	reference to the object
`(string -> string)map`	value	New value to set

Minimum Role: pool-admin

Return Type: `void`

RPC name: unplug *Overview:*

Unplug the vusb device from the vm.

Signature:

```
void unplug (session ref session_id, VUSB ref self)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
`VUSB ref`	self	vusb deivce

Minimum Role: pool-admin

Return Type: `void`

API Reference - Error Handling

January 23, 2024

When a low-level transport error occurs, or a request is malformed at the HTTP or RPC level, the server may send an HTTP 500 error response, or the client may simulate the same. The client must be prepared to handle these errors, though they may be treated as fatal.

On the wire, these are transmitted in a form similar to this when using the
XML-RPC protocol:

```
$curl -D - -X POST https://server -H 'Content-Type: application/xml' \
> -d '<?xml version="1.0"?>
> <methodCall>
>   <methodName>session.logout</methodName>
> </methodCall>'
HTTP/1.1 500 Internal Error
content-length: 297
content-type:text/html
connection:close
cache-control:no-cache, no-store

<html><body><h1>HTTP 500 internal server error</h1>An unexpected error
    occurred;
 please wait a while and try again. If the problem persists, please
     contact your
 support representative.<h1> Additional information </h1>Xmlrpc.
     Parse_error(&quo
t;close_tag&quot;, &quot;open_tag&quot;, _)</body></html>
<!--NeedCopy-->
```

When using the JSON-RPC protocol:

```
$curl -D - -X POST https://server/jsonrpc -H 'Content-Type: application
    /json' \
> -d '{

>      "jsonrpc": "2.0",
>      "method": "session.login_with_password",
>      "id": 0
>   }
 '
HTTP/1.1 500 Internal Error
content-length: 308
content-type:text/html
connection:close
cache-control:no-cache, no-store

<html><body><h1>HTTP 500 internal server error</h1>An unexpected error
    occurred;
 please wait a while and try again. If the problem persists, please
     contact your
 support representative.<h1> Additional information </h1>Jsonrpc.
     Malformed_metho
d_request(&quot;{
 jsonrpc=...,method=...,id=... }
 &quot;)</body></html>
<!--NeedCopy-->
```

All other failures are reported with a more structured error response, to
allow better automatic response to failures, proper internationalisation of

any error message, and easier debugging.

On the wire, these are transmitted like this when using the XML-RPC protocol:

```
    <struct>
      <member>
        <name>Status</name>
        <value>Failure</value>
      </member>
      <member>
        <name>ErrorDescription</name>
        <value>
          <array>
            <data>
              <value>MAP_DUPLICATE_KEY</value>
              <value>Customer</value>
              <value>eSpiel Inc.</value>
              <value>eSpiel Incorporated</value>
            </data>
          </array>
        </value>
      </member>
    </struct>
<!--NeedCopy-->
```

Note that `ErrorDescription` value is an array of string values. The first element of the array is an error code; the remainder of the array are strings representing error parameters relating to that code. In this case, the client has attempted to add the mapping *Customer -> eSpiel Incorporated* to a Map, but it already contains the mapping *Customer -> eSpiel Inc.*, and so the request has failed.

When using the JSON-RPC protocol v2.0, the above error is transmitted as:

```
{

    "jsonrpc": "2.0",
    "error": {

        "code": 1,
        "message": "MAP_DUPLICATE_KEY",
        "data": [
            "Customer","eSpiel Inc.","eSpiel Incorporated"
        ]
     }
 ,
    "id": 3
   }

<!--NeedCopy-->
```

Finally, when using the JSON-RPC protocol v1.0:

```
{

  "result": null,
  "error": [
      "MAP_DUPLICATE_KEY","Customer","eSpiel Inc.","eSpiel Incorporated
          "
  ],
  "id": "xyz"
 }

<!--NeedCopy-->
```

Each possible error code is documented in the following section.

Error Codes

ACTIVATION_WHILE_NOT_FREE

An activation key can only be applied when the edition is set to 'free'.

No parameters.

ADDRESS_VIOLATES_LOCKING_CONSTRAINT

The specified IP address violates the VIF locking configuration.

Signature:

```
ADDRESS_VIOLATES_LOCKING_CONSTRAINT(address)
<!--NeedCopy-->
```

APPLY_GUIDANCE_FAILED

Failed to apply guidance on a host after updating.

Signature:

```
APPLY_GUIDANCE_FAILED(ref)
<!--NeedCopy-->
```

APPLY_LIVEPATCH_FAILED

Failed to apply a livepatch.

Signature:

```
APPLY_LIVEPATCH_FAILED(livepatch)
<!--NeedCopy-->
```

APPLY_UPDATES_FAILED

Failed to apply updates on a host.

Signature:

```
APPLY_UPDATES_FAILED(ref)
<!--NeedCopy-->
```

APPLY_UPDATES_IN_PROGRESS

The operation could not be performed because applying updates is in progress.

No parameters.

AUTH_ALREADY_ENABLED

External authentication for this server is already enabled.

Signature:

```
AUTH_ALREADY_ENABLED(current auth_type, current service_name)
<!--NeedCopy-->
```

AUTH_DISABLE_FAILED

The host failed to disable external authentication.

Signature:

```
AUTH_DISABLE_FAILED(message)
<!--NeedCopy-->
```

AUTH_DISABLE_FAILED_PERMISSION_DENIED

The host failed to disable external authentication.

Signature:

```
AUTH_DISABLE_FAILED_PERMISSION_DENIED(message)
<!--NeedCopy-->
```

AUTH_DISABLE_FAILED_WRONG_CREDENTIALS

The host failed to disable external authentication.

Signature:

```
AUTH_DISABLE_FAILED_WRONG_CREDENTIALS(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED_DOMAIN_LOOKUP_FAILED

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED_DOMAIN_LOOKUP_FAILED(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED_INVALID_ACCOUNT

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED_INVALID_ACCOUNT(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED_INVALID_OU

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED_INVALID_OU(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED_PERMISSION_DENIED

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED_PERMISSION_DENIED(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED_UNAVAILABLE

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED_UNAVAILABLE(message)
<!--NeedCopy-->
```

AUTH_ENABLE_FAILED_WRONG_CREDENTIALS

The host failed to enable external authentication.

Signature:

```
AUTH_ENABLE_FAILED_WRONG_CREDENTIALS(message)
<!--NeedCopy-->
```

AUTH_IS_DISABLED

External authentication is disabled, unable to resolve subject name.

No parameters.

AUTH_SERVICE_ERROR

Error querying the external directory service.

Signature:

```
AUTH_SERVICE_ERROR(message)
<!--NeedCopy-->
```

AUTH_UNKNOWN_TYPE

Unknown type of external authentication.

Signature:

```
AUTH_UNKNOWN_TYPE(type)
<!--NeedCopy-->
```

BACKUP_SCRIPT_FAILED

The backup could not be performed because the backup script failed.

Signature:

```
BACKUP_SCRIPT_FAILED(log)
<!--NeedCopy-->
```

BALLOONING_TIMEOUT_BEFORE_MIGRATION

Timeout trying to balloon down memory before VM migration. If the error occurs repeatedly, consider increasing the memory-dynamic-min value.

Signature:

```
BALLOONING_TIMEOUT_BEFORE_MIGRATION(vm)
<!--NeedCopy-->
```

BOOTLOADER_FAILED

The bootloader returned an error

Signature:

```
BOOTLOADER_FAILED(vm, msg)
<!--NeedCopy-->
```

BRIDGE_NAME_EXISTS

The specified bridge already exists.

Signature:

```
BRIDGE_NAME_EXISTS(bridge)
<!--NeedCopy-->
```

BRIDGE_NOT_AVAILABLE

Could not find bridge required by VM.

Signature:

```
BRIDGE_NOT_AVAILABLE(bridge)
<!--NeedCopy-->
```

CANNOT_ADD_TUNNEL_TO_BOND_SLAVE

This PIF is a bond member and cannot have a tunnel on it.

Signature:

```
CANNOT_ADD_TUNNEL_TO_BOND_SLAVE(PIF)
<!--NeedCopy-->
```

CANNOT_ADD_TUNNEL_TO_SRIOV_LOGICAL

This is a network SR-IOV logical PIF and cannot have a tunnel on it.

Signature:

```
CANNOT_ADD_TUNNEL_TO_SRIOV_LOGICAL(PIF)
<!--NeedCopy-->
```

CANNOT_ADD_TUNNEL_TO_VLAN_ON_SRIOV_LOGICAL

This is a vlan PIF on network SR-IOV and cannot have a tunnel on it.

Signature:

```
CANNOT_ADD_TUNNEL_TO_VLAN_ON_SRIOV_LOGICAL(PIF)
<!--NeedCopy-->
```

CANNOT_ADD_VLAN_TO_BOND_SLAVE

This PIF is a bond member and cannot have a VLAN on it.

Signature:

```
CANNOT_ADD_VLAN_TO_BOND_SLAVE(PIF)
<!--NeedCopy-->
```

CANNOT_CHANGE_PIF_PROPERTIES

The properties of this PIF cannot be changed. Only the properties of non-bonded physical PIFs, or bond interfaces can be changed.

Signature:

```
CANNOT_CHANGE_PIF_PROPERTIES(PIF)
<!--NeedCopy-->
```

CANNOT_CONTACT_HOST

Cannot forward messages because the server cannot be contacted. The server may be switched off or there may be network connectivity problems.

Signature:

```
CANNOT_CONTACT_HOST(host)
<!--NeedCopy-->
```

CANNOT_CREATE_STATE_FILE

An HA statefile could not be created, perhaps because no SR with the appropriate capability was found.

No parameters.

CANNOT_DESTROY_DISASTER_RECOVERY_TASK

The disaster recovery task could not be cleanly destroyed.

Signature:

```
CANNOT_DESTROY_DISASTER_RECOVERY_TASK(reason)
<!--NeedCopy-->
```

CANNOT_DESTROY_SYSTEM_NETWORK

You tried to destroy a system network: these cannot be destroyed.

Signature:

```
CANNOT_DESTROY_SYSTEM_NETWORK(network)
<!--NeedCopy-->
```

CANNOT_ENABLE_REDO_LOG

Could not enable redo log.

Signature:

```
CANNOT_ENABLE_REDO_LOG(reason)
<!--NeedCopy-->
```

CANNOT_EVACUATE_HOST

This server cannot be evacuated.

Signature:

```
CANNOT_EVACUATE_HOST(errors)
<!--NeedCopy-->
```

CANNOT_FETCH_PATCH

The requested update could not be obtained from the coordinator.

Signature:

```
CANNOT_FETCH_PATCH(uuid)
<!--NeedCopy-->
```

CANNOT_FIND_OEM_BACKUP_PARTITION

The backup partition to stream the update to cannot be found.

No parameters.

CANNOT_FIND_PATCH

The requested update could not be found. This can occur when you designate a new coordinator or xe patch-clean. Please upload the update again.

No parameters.

CANNOT_FIND_STATE_PARTITION

This operation could not be performed because the state partition could not be found

No parameters.

CANNOT_FIND_UPDATE

The requested update could not be found. Please upload the update again. This can occur when you run xe update-pool-clean before xe update-apply.

No parameters.

CANNOT_FORGET_SRIOV_LOGICAL

This is a network SR-IOV logical PIF and cannot do forget on it

Signature:

```
CANNOT_FORGET_SRIOV_LOGICAL(PIF)
<!--NeedCopy-->
```

CANNOT_PLUG_BOND_SLAVE

This PIF is a bond member and cannot be plugged.

Signature:

```
CANNOT_PLUG_BOND_SLAVE(PIF)
<!--NeedCopy-->
```

CANNOT_PLUG_VIF

Cannot plug VIF

Signature:

```
CANNOT_PLUG_VIF(VIF)
<!--NeedCopy-->
```

CANNOT_RESET_CONTROL_DOMAIN

The power-state of a control domain cannot be reset.

Signature:

```
CANNOT_RESET_CONTROL_DOMAIN(vm)
<!--NeedCopy-->
```

CANNOT_RESTART_DEVICE_MODEL

Cannot restart device models of paused VMs residing on the host.

Signature:

```
CANNOT_RESTART_DEVICE_MODEL(ref)
<!--NeedCopy-->
```

CERTIFICATE_ALREADY_EXISTS

A certificate already exists with the specified name.

Signature:

```
CERTIFICATE_ALREADY_EXISTS(name)
<!--NeedCopy-->
```

CERTIFICATE_CORRUPT

The specified certificate is corrupt or unreadable.

Signature:

```
CERTIFICATE_CORRUPT(name)
<!--NeedCopy-->
```

CERTIFICATE_DOES_NOT_EXIST

The specified certificate does not exist.

Signature:

```
CERTIFICATE_DOES_NOT_EXIST(name)
<!--NeedCopy-->
```

CERTIFICATE_LIBRARY_CORRUPT

The certificate library is corrupt or unreadable.

No parameters.

CERTIFICATE_NAME_INVALID

The specified certificate name is invalid.

Signature:

```
CERTIFICATE_NAME_INVALID(name)
<!--NeedCopy-->
```

CHANGE_PASSWORD_REJECTED

The system rejected the password change request; perhaps the new password was too short?

Signature:

```
CHANGE_PASSWORD_REJECTED(msg)
<!--NeedCopy-->
```

CLUSTERED_SR_DEGRADED

An SR is using clustered local storage. It is not safe to reboot a host at the moment.

Signature:

```
CLUSTERED_SR_DEGRADED(sr)
<!--NeedCopy-->
```

CLUSTERING_DISABLED

An operation was attempted while clustering was disabled on the cluster_host.

Signature:

```
CLUSTERING_DISABLED(cluster_host)
<!--NeedCopy-->
```

CLUSTERING_ENABLED

An operation was attempted while clustering was enabled on the cluster_host.

Signature:

```
CLUSTERING_ENABLED(cluster_host)
<!--NeedCopy-->
```

CLUSTER_ALREADY_EXISTS

A cluster already exists in the pool.

No parameters.

CLUSTER_CREATE_IN_PROGRESS

The operation could not be performed because cluster creation is in progress.

No parameters.

CLUSTER_DOES_NOT_HAVE_ONE_NODE

An operation failed as it expected the cluster to have only one node but found multiple cluster_hosts.

Signature:

```
CLUSTER_DOES_NOT_HAVE_ONE_NODE(number_of_nodes)
<!--NeedCopy-->
```

CLUSTER_FORCE_DESTROY_FAILED

Force destroy failed on a Cluster_host while force destroying the cluster.

Signature:

```
CLUSTER_FORCE_DESTROY_FAILED(cluster)
<!--NeedCopy-->
```

CLUSTER_HOST_IS_LAST

The last cluster host cannot be destroyed. Destroy the cluster instead

Signature:

```
CLUSTER_HOST_IS_LAST(cluster_host)
<!--NeedCopy-->
```

CLUSTER_HOST_NOT_JOINED

Cluster_host operation failed as the cluster_host has not joined the cluster.

Signature:

```
CLUSTER_HOST_NOT_JOINED(cluster_host)
<!--NeedCopy-->
```

CLUSTER_STACK_IN_USE

The cluster stack is still in use by at least one plugged PBD.

Signature:

```
CLUSTER_STACK_IN_USE(cluster_stack)
<!--NeedCopy-->
```

CONFIGURE_REPOSITORIES_IN_PROGRESS

The operation could not be performed because other repository(ies) is(are) already being configured.

No parameters.

COULD_NOT_FIND_NETWORK_INTERFACE_WITH_SPECIFIED_DEVICE_NAME_AND_MAC_ADDRESS

Could not find a network interface with the specified device name and MAC address.

Signature:

```
COULD_NOT_FIND_NETWORK_INTERFACE_WITH_SPECIFIED_DEVICE_NAME_AND_MAC_ADDRESS
    (device, mac)
<!--NeedCopy-->
```

COULD_NOT_IMPORT_DATABASE

An error occurred while attempting to import a database from a metadata VDI

Signature:

```
COULD_NOT_IMPORT_DATABASE(reason)
<!--NeedCopy-->
```

COULD_NOT_UPDATE_IGMP_SNOOPING_EVERYWHERE

The IGMP Snooping setting cannot be applied for some of the host, network(s).

No parameters.

CPU_FEATURE_MASKING_NOT_SUPPORTED

The CPU does not support masking of features.

Signature:

```
CPU_FEATURE_MASKING_NOT_SUPPORTED(details)
<!--NeedCopy-->
```

CRL_ALREADY_EXISTS

A CRL already exists with the specified name.

Signature:

```
CRL_ALREADY_EXISTS(name)
<!--NeedCopy-->
```

CRL_CORRUPT

The specified CRL is corrupt or unreadable.

Signature:

```
CRL_CORRUPT(name)
<!--NeedCopy-->
```

CRL_DOES_NOT_EXIST

The specified CRL does not exist.

Signature:

```
CRL_DOES_NOT_EXIST(name)
<!--NeedCopy-->
```

CRL_NAME_INVALID

The specified CRL name is invalid.

Signature:

```
CRL_NAME_INVALID(name)
<!--NeedCopy-->
```

DB_UNIQUENESS_CONSTRAINT_VIOLATION

You attempted an operation which would have resulted in duplicate keys in the database.

Signature:

```
DB_UNIQUENESS_CONSTRAINT_VIOLATION(table, field, value)
<!--NeedCopy-->
```

DEFAULT_SR_NOT_FOUND

The default SR reference does not point to a valid SR

Signature:

```
DEFAULT_SR_NOT_FOUND(sr)
<!--NeedCopy-->
```

DEVICE_ALREADY_ATTACHED

The device is already attached to a VM

Signature:

```
DEVICE_ALREADY_ATTACHED(device)
<!--NeedCopy-->
```

DEVICE_ALREADY_DETACHED

The device is not currently attached

Signature:

```
DEVICE_ALREADY_DETACHED(device)
<!--NeedCopy-->
```

DEVICE_ALREADY_EXISTS

A device with the name given already exists on the selected VM

Signature:

```
DEVICE_ALREADY_EXISTS(device)
<!--NeedCopy-->
```

DEVICE_ATTACH_TIMEOUT

A timeout happened while attempting to attach a device to a VM.

Signature:

```
DEVICE_ATTACH_TIMEOUT(type, ref)
<!--NeedCopy-->
```

DEVICE_DETACH_REJECTED

The VM rejected the attempt to detach the device.

Signature:

```
DEVICE_DETACH_REJECTED(type, ref, msg)
<!--NeedCopy-->
```

DEVICE_DETACH_TIMEOUT

A timeout happened while attempting to detach a device from a VM.

Signature:

```
DEVICE_DETACH_TIMEOUT(type, ref)
<!--NeedCopy-->
```

DEVICE_NOT_ATTACHED

The operation could not be performed because the VBD was not connected to the VM.

Signature:

```
DEVICE_NOT_ATTACHED(VBD)
<!--NeedCopy-->
```

DISK_VBD_MUST_BE_READWRITE_FOR_HVM

All VBDs of type 'disk' must be read/write for HVM guests

Signature:

```
DISK_VBD_MUST_BE_READWRITE_FOR_HVM(vbd)
<!--NeedCopy-->
```

DOMAIN_BUILDER_ERROR

An internal error generated by the domain builder.

Signature:

```
DOMAIN_BUILDER_ERROR(function, code, message)
<!--NeedCopy-->
```

DOMAIN_EXISTS

The operation could not be performed because a domain still exists for the specified VM.

Signature:

```
DOMAIN_EXISTS(vm, domid)
<!--NeedCopy-->
```

DUPLICATE_MAC_SEED

This MAC seed is already in use by a VM in the pool

Signature:

```
DUPLICATE_MAC_SEED(seed)
<!--NeedCopy-->
```

DUPLICATE_PIF_DEVICE_NAME

A PIF with this specified device name already exists.

Signature:

```
DUPLICATE_PIF_DEVICE_NAME(device)
<!--NeedCopy-->
```

DUPLICATE_VM

Cannot restore this VM because it would create a duplicate

Signature:

```
DUPLICATE_VM(vm)
<!--NeedCopy-->
```

EVENTS_LOST

Some events have been lost from the queue and cannot be retrieved.

No parameters.

EVENT_FROM_TOKEN_PARSE_FAILURE

The event.from token could not be parsed. Valid values include: ‘’, and a value returned from a previous event.from call.

Signature:

```
EVENT_FROM_TOKEN_PARSE_FAILURE(token)
<!--NeedCopy-->
```

EVENT_SUBSCRIPTION_PARSE_FAILURE

The server failed to parse your event subscription. Valid values include: *, class-name, class-name/object-reference.

Signature:

```
EVENT_SUBSCRIPTION_PARSE_FAILURE(subscription)
<!--NeedCopy-->
```

FAILED_TO_START_EMULATOR

An emulator required to run this VM failed to start

Signature:

```
FAILED_TO_START_EMULATOR(vm, name, msg)
<!--NeedCopy-->
```

FEATURE_REQUIRES_HVM

The VM is set up to use a feature that requires it to boot as HVM.

Signature:

```
FEATURE_REQUIRES_HVM(details)
<!--NeedCopy-->
```

FEATURE_RESTRICTED

The use of this feature is restricted.

No parameters.

FIELD_TYPE_ERROR

The value specified is of the wrong type

Signature:

```
FIELD_TYPE_ERROR(field)
<!--NeedCopy-->
```

GET_HOST_UPDATES_FAILED

Failed to get available updates from a host.

Signature:

```
GET_HOST_UPDATES_FAILED(ref)
<!--NeedCopy-->
```

GET_UPDATES_FAILED

Failed to get available updates from the pool.

No parameters.

GET_UPDATES_IN_PROGRESS

The operation could not be performed because getting updates is in progress.

No parameters.

GPU_GROUP_CONTAINS_NO_PGPUS

The GPU group does not contain any PGPUs.

Signature:

```
GPU_GROUP_CONTAINS_NO_PGPUS(gpu_group)
<!--NeedCopy-->
```

GPU_GROUP_CONTAINS_PGPU

The GPU group contains active PGPUs and cannot be deleted.

Signature:

```
GPU_GROUP_CONTAINS_PGPU(pgpus)
<!--NeedCopy-->
```

GPU_GROUP_CONTAINS_VGPU

The GPU group contains active VGPUs and cannot be deleted.

Signature:

```
GPU_GROUP_CONTAINS_VGPU(vgpus)
<!--NeedCopy-->
```

HANDLE_INVALID

You gave an invalid object reference. The object may have recently been deleted. The class parameter gives the type of reference given, and the handle parameter echoes the bad value given.

Signature:

```
HANDLE_INVALID(class, handle)
<!--NeedCopy-->
```

HA_ABORT_NEW_MASTER

This server cannot accept the proposed new coordinator setting at this time.

Signature:

```
HA_ABORT_NEW_MASTER(reason)
<!--NeedCopy-->
```

HA_CANNOT_CHANGE_BOND_STATUS_OF_MGMT_IFACE

This operation cannot be performed because creating or deleting a bond involving the management interface is not allowed while HA is on. In order to do that, disable HA, create or delete the bond then re-enable HA.

No parameters.

HA_CONSTRAINT_VIOLATION_NETWORK_NOT_SHARED

This operation cannot be performed because the referenced network is not properly shared. The network must either be entirely virtual or must be physically present via a currently_attached PIF on every host.

Signature:

```
HA_CONSTRAINT_VIOLATION_NETWORK_NOT_SHARED(network)
<!--NeedCopy-->
```

HA_CONSTRAINT_VIOLATION_SR_NOT_SHARED

This operation cannot be performed because the referenced SR is not properly shared. The SR must both be marked as shared and a currently_attached PBD must exist for each host.

Signature:

```
HA_CONSTRAINT_VIOLATION_SR_NOT_SHARED(SR)
<!--NeedCopy-->
```

HA_DISABLE_IN_PROGRESS

The operation could not be performed because HA disable is in progress

No parameters.

HA_ENABLE_IN_PROGRESS

The operation could not be performed because HA enable is in progress

No parameters.

HA_FAILED_TO_FORM_LIVESET

HA could not be enabled on the Pool because a liveset could not be formed: check storage and network heartbeat paths.

No parameters.

HA_HEARTBEAT_DAEMON_STARTUP_FAILED

The server could not join the liveset because the HA daemon failed to start.

No parameters.

HA_HOST_CANNOT_ACCESS_STATEFILE

The server could not join the liveset because the HA daemon could not access the heartbeat disk.

No parameters.

HA_HOST_CANNOT_SEE_PEERS

The operation failed because the HA software on the specified server could not see a subset of other servers. Check your network connectivity.

Signature:

```
HA_HOST_CANNOT_SEE_PEERS(host, all, subset)
<!--NeedCopy-->
```

HA_HOST_IS_ARMED

The operation could not be performed while the server is still armed; it must be disarmed first.

Signature:

```
HA_HOST_IS_ARMED(host)
<!--NeedCopy-->
```

HA_IS_ENABLED

The operation could not be performed because HA is enabled on the Pool

No parameters.

HA_LOST_STATEFILE

This server lost access to the HA statefile.

No parameters.

HA_NOT_ENABLED

The operation could not be performed because HA is not enabled on the Pool

No parameters.

HA_NOT_INSTALLED

The operation could not be performed because the HA software is not installed on this server.

Signature:

```
HA_NOT_INSTALLED(host)
<!--NeedCopy-->
```

HA_NO_PLAN

Cannot find a plan for placement of VMs as there are no other servers available.

No parameters.

HA_OPERATION_WOULD_BREAK_FAILOVER_PLAN

This operation cannot be performed because it would invalidate VM failover planning such that the system would be unable to guarantee to restart protected VMs after a Host failure.

No parameters.

HA_POOL_IS_ENABLED_BUT_HOST_IS_DISABLED

This server cannot join the pool because the pool has HA enabled but this server has HA disabled.

No parameters.

HA_SHOULD_BE_FENCED

Server cannot rejoin pool because it should have fenced (it is not in the coordinator's partition).

Signature:

```
HA_SHOULD_BE_FENCED(host)
<!--NeedCopy-->
```

HA_TOO_FEW_HOSTS

HA can only be enabled for 2 servers or more. Note that 2 servers requires a pre-configured quorum tiebreak script.

No parameters.

HOSTS_NOT_COMPATIBLE

The hosts in this pool are not compatible.

No parameters.

HOSTS_NOT_HOMOGENEOUS

The hosts in this pool are not homogeneous.

Signature:

```
HOSTS_NOT_HOMOGENEOUS(reason)
<!--NeedCopy-->
```

HOST_BROKEN

This server failed in the middle of an automatic failover operation and needs to retry the failover action.

No parameters.

HOST_CANNOT_ATTACH_NETWORK

Server cannot attach network (in the case of NIC bonding, this may be because attaching the network on this server would require other networks - that are currently active - to be taken down).

Signature:

```
HOST_CANNOT_ATTACH_NETWORK(host, network)
<!--NeedCopy-->
```

HOST_CANNOT_DESTROY_SELF

The pool coordinator host cannot be removed.

Signature:

```
HOST_CANNOT_DESTROY_SELF(host)
<!--NeedCopy-->
```

HOST_CANNOT_READ_METRICS

The metrics of this server could not be read.

No parameters.

HOST_CD_DRIVE_EMPTY

The host CDROM drive does not contain a valid CD

No parameters.

HOST_DISABLED

The specified server is disabled.

Signature:

```
HOST_DISABLED(host)
<!--NeedCopy-->
```

HOST_DISABLED_UNTIL_REBOOT

The specified server is disabled and cannot be re-enabled until after it has rebooted.

Signature:

```
HOST_DISABLED_UNTIL_REBOOT(host)
<!--NeedCopy-->
```

HOST_EVACUATE_IN_PROGRESS

This host is being evacuated.

Signature:

```
HOST_EVACUATE_IN_PROGRESS(host)
<!--NeedCopy-->
```

HOST_HAS_NO_MANAGEMENT_IP

The server failed to acquire an IP address on its management interface and therefore cannot contact the coordinator.

No parameters.

HOST_HAS_RESIDENT_VMS

This server cannot be forgotten because there are user VMs still running.

Signature:

```
HOST_HAS_RESIDENT_VMS(host)
<!--NeedCopy-->
```

HOST_IN_EMERGENCY_MODE

Cannot perform operation as the host is running in emergency mode.

No parameters.

HOST_IN_USE

This operation cannot be completed as the host is in use by (at least) the object of type and ref echoed below.

Signature:

```
HOST_IN_USE(host, type, ref)
<!--NeedCopy-->
```

HOST_IS_LIVE

This operation cannot be completed because the server is still live.

Signature:

```
HOST_IS_LIVE(host)
<!--NeedCopy-->
```

HOST_IS_SLAVE

You cannot make regular API calls directly on a supporter. Please pass API calls via the coordinator host.

Signature:

```
HOST_IS_SLAVE(Master IP address)
<!--NeedCopy-->
```

HOST_ITS_OWN_SLAVE

The host is its own supporter. Please use pool-emergency-transition-to-master or pool-emergency-reset-master.

No parameters.

HOST_MASTER_CANNOT_TALK_BACK

The coordinator reports that it cannot talk back to the supporter on the supplied management IP address.

Signature:

```
HOST_MASTER_CANNOT_TALK_BACK(ip)
<!--NeedCopy-->
```

HOST_NAME_INVALID

The server name is invalid.

Signature:

```
HOST_NAME_INVALID(reason)
<!--NeedCopy-->
```

HOST_NOT_DISABLED

This operation cannot be performed because the host is not disabled. Please disable the host and then try again.

No parameters.

HOST_NOT_ENOUGH_FREE_MEMORY

Not enough server memory is available to perform this operation.

Signature:

```
HOST_NOT_ENOUGH_FREE_MEMORY(needed, available)
<!--NeedCopy-->
```

HOST_NOT_ENOUGH_PCPUS

The host does not have enough pCPUs to run the VM. It needs at least as many as the VM has vCPUs.

Signature:

```
HOST_NOT_ENOUGH_PCPUS(vcpus, pcpus)
<!--NeedCopy-->
```

HOST_NOT_LIVE

This operation cannot be completed as the server is not live.

No parameters.

HOST_OFFLINE

You attempted an operation which involves a host which could not be contacted.

Signature:

```
HOST_OFFLINE(host)
<!--NeedCopy-->
```

HOST_POWER_ON_MODE_DISABLED

This operation cannot be completed because the server power on mode is disabled.

No parameters.

HOST_STILL_BOOTING

The host toolstack is still initialising. Please wait.

No parameters.

HOST_UNKNOWN_TO_MASTER

The coordinator says the server is not known to it. Is the server in the coordinator's database and pointing to the correct coordinator? Are all servers using the same pool secret?

Signature:

```
HOST_UNKNOWN_TO_MASTER(host)
<!--NeedCopy-->
```

HOST_XAPI_VERSION_HIGHER_THAN_COORDINATOR

The host xapi version is higher than the one in the coordinator

Signature:

```
HOST_XAPI_VERSION_HIGHER_THAN_COORDINATOR(host_xapi_version)
<!--NeedCopy-->
```

ILLEGAL_VBD_DEVICE

The specified VBD device is not recognized: please use a non-negative integer

Signature:

```
ILLEGAL_VBD_DEVICE(vbd, device)
<!--NeedCopy-->
```

IMPORT_ERROR

The VM could not be imported.

Signature:

```
IMPORT_ERROR(msg)
<!--NeedCopy-->
```

IMPORT_ERROR_ATTACHED_DISKS_NOT_FOUND

The VM could not be imported because attached disks could not be found.

No parameters.

IMPORT_ERROR_CANNOT_HANDLE_CHUNKED

Cannot import VM using chunked encoding.

No parameters.

IMPORT_ERROR_FAILED_TO_FIND_OBJECT

The VM could not be imported because a required object could not be found.

Signature:

```
IMPORT_ERROR_FAILED_TO_FIND_OBJECT(id)
<!--NeedCopy-->
```

IMPORT_ERROR_PREMATURE_EOF

The VM could not be imported; the end of the file was reached prematurely.

No parameters.

IMPORT_ERROR_SOME_CHECKSUMS_FAILED

Some data checksums were incorrect; the VM may be corrupt.

No parameters.

IMPORT_ERROR_UNEXPECTED_FILE

The VM could not be imported because the XVA file is invalid: an unexpected file was encountered.

Signature:

```
IMPORT_ERROR_UNEXPECTED_FILE(filename_expected, filename_found)
<!--NeedCopy-->
```

IMPORT_INCOMPATIBLE_VERSION

The import failed because this export has been created by a different (incompatible) product version

No parameters.

INCOMPATIBLE_CLUSTER_STACK_ACTIVE

This operation cannot be performed, because it is incompatible with the currently active HA cluster stack.

Signature:

```
INCOMPATIBLE_CLUSTER_STACK_ACTIVE(cluster_stack)
<!--NeedCopy-->
```

INCOMPATIBLE_PIF_PROPERTIES

These PIFs cannot be bonded, because their properties are different.

No parameters.

INCOMPATIBLE_STATEFILE_SR

The specified SR is incompatible with the selected HA cluster stack.

Signature:

```
INCOMPATIBLE_STATEFILE_SR(SR type)
<!--NeedCopy-->
```

INTERFACE_HAS_NO_IP

The specified interface cannot be used because it has no IP address

Signature:

```
INTERFACE_HAS_NO_IP(interface)
<!--NeedCopy-->
```

INTERNAL_ERROR

The server failed to handle your request, due to an internal error. The given message may give details useful for debugging the problem.

Signature:

```
INTERNAL_ERROR(message)
<!--NeedCopy-->
```

INVALID_BASE_URL

The base url in the repository is invalid.

Signature:

```
INVALID_BASE_URL(url)
<!--NeedCopy-->
```

INVALID_CIDR_ADDRESS_SPECIFIED

A required parameter contained an invalid CIDR address (<addr>/<prefix length>)

Signature:

```
INVALID_CIDR_ADDRESS_SPECIFIED(parameter)
<!--NeedCopy-->
```

INVALID_CLUSTER_STACK

The cluster stack provided is not supported.

Signature:

```
INVALID_CLUSTER_STACK(cluster_stack)
<!--NeedCopy-->
```

INVALID_DEVICE

The device name is invalid

Signature:

```
INVALID_DEVICE(device)
<!--NeedCopy-->
```

INVALID_EDITION

The edition you supplied is invalid.

Signature:

```
INVALID_EDITION(edition)
<!--NeedCopy-->
```

INVALID_FEATURE_STRING

The given feature string is not valid.

Signature:

```
INVALID_FEATURE_STRING(details)
<!--NeedCopy-->
```

INVALID_GPGKEY_PATH

The GPG public key file name in the repository is invalid.

Signature:

```
INVALID_GPGKEY_PATH(gpgkey_path)
<!--NeedCopy-->
```

INVALID_IP_ADDRESS_SPECIFIED

A required parameter contained an invalid IP address

Signature:

```
INVALID_IP_ADDRESS_SPECIFIED(parameter)
<!--NeedCopy-->
```

INVALID_PATCH

The uploaded patch file is invalid

No parameters.

INVALID_PATCH_WITH_LOG

The uploaded patch file is invalid. See attached log for more details.

Signature:

```
INVALID_PATCH_WITH_LOG(log)
<!--NeedCopy-->
```

INVALID_REPOMD_XML

The repomd.xml is invalid.

No parameters.

INVALID_REPOSITORY_DOMAIN_ALLOWLIST

The repository domain allowlist has some invalid domains.

Signature:

```
INVALID_REPOSITORY_DOMAIN_ALLOWLIST(domains)
<!--NeedCopy-->
```

INVALID_REPOSITORY_PROXY_CREDENTIAL

The repository proxy username/password is invalid.

No parameters.

INVALID_REPOSITORY_PROXY_URL

The repository proxy URL is invalid.

Signature:

```
INVALID_REPOSITORY_PROXY_URL(url)
<!--NeedCopy-->
```

INVALID_UPDATE

The uploaded update package is invalid.

Signature:

```
INVALID_UPDATE(info)
<!--NeedCopy-->
```

INVALID_UPDATEINFO_XML

The updateinfo.xml is invalid.

No parameters.

INVALID_UPDATE_SYNC_DAY

Invalid day of the week chosen for weekly update sync.

Signature:

```
INVALID_UPDATE_SYNC_DAY(day)
<!--NeedCopy-->
```

INVALID_VALUE

The value given is invalid

Signature:

```
INVALID_VALUE(field, value)
<!--NeedCopy-->
```

IS_TUNNEL_ACCESS_PIF

Cannot create a VLAN or tunnel on top of a tunnel access PIF - use the underlying transport PIF instead.

Signature:

```
IS_TUNNEL_ACCESS_PIF(PIF)
<!--NeedCopy-->
```

JOINING_HOST_CANNOT_BE_MASTER_OF_OTHER_HOSTS

The server joining the pool cannot already be a coordinator of another pool.

No parameters.

JOINING_HOST_CANNOT_CONTAIN_SHARED_SRS

The server joining the pool cannot contain any shared storage.

No parameters.

JOINING_HOST_CANNOT_HAVE_RUNNING_OR_SUSPENDED_VMS

The server joining the pool cannot have any running or suspended VMs.

No parameters.

JOINING_HOST_CANNOT_HAVE_RUNNING_VMS

The server joining the pool cannot have any running VMs.

No parameters.

JOINING_HOST_CANNOT_HAVE_VMS_WITH_CURRENT_OPERATIONS

The host joining the pool cannot have any VMs with active tasks.

No parameters.

JOINING_HOST_CONNECTION_FAILED

There was an error connecting to the host while joining it in the pool.

No parameters.

JOINING_HOST_SERVICE_FAILED

There was an error connecting to the server. The service contacted didn't reply properly.

No parameters.

LICENCE_RESTRICTION

This operation is not allowed because your license lacks a needed feature. Please contact your support representative.

Signature:

```
LICENCE_RESTRICTION(feature)
<!--NeedCopy-->
```

LICENSE_CANNOT_DOWNGRADE_WHILE_IN_POOL

Cannot downgrade license while in pool. Please disband the pool first, then downgrade licenses on hosts separately.

No parameters.

LICENSE_CHECKOUT_ERROR

The license for the edition you requested is not available.

Signature:

```
LICENSE_CHECKOUT_ERROR(reason)
<!--NeedCopy-->
```

LICENSE_DOES_NOT_SUPPORT_POOLING

This server cannot join a pool because its license does not support pooling.

No parameters.

LICENSE_DOES_NOT_SUPPORT_XHA

HA cannot be enabled because this server's license does not allow it.

No parameters.

LICENSE_EXPIRED

Your license has expired. Please contact your support representative.

No parameters.

LICENSE_FILE_DEPRECATED

This type of license file is for previous versions of the server. Please upgrade to the new licensing system.

No parameters.

LICENSE_HOST_POOL_MISMATCH

Host and pool have incompatible licenses (editions).

No parameters.

LICENSE_PROCESSING_ERROR

There was an error processing your license. Please contact your support representative.

No parameters.

LOCATION_NOT_UNIQUE

A VDI with the specified location already exists within the SR

Signature:

```
LOCATION_NOT_UNIQUE(SR, location)
<!--NeedCopy-->
```

MAC_DOES_NOT_EXIST

The MAC address specified does not exist on this server.

Signature:

```
MAC_DOES_NOT_EXIST(MAC)
<!--NeedCopy-->
```

MAC_INVALID

The MAC address specified is not valid.

Signature:

```
MAC_INVALID(MAC)
<!--NeedCopy-->
```

MAC_STILL_EXISTS

The MAC address specified still exists on this server.

Signature:

```
MAC_STILL_EXISTS(MAC)
<!--NeedCopy-->
```

MAP_DUPLICATE_KEY

You tried to add a key-value pair to a map, but that key is already there.

Signature:

```
MAP_DUPLICATE_KEY(type, param_name, uuid, key)
<!--NeedCopy-->
```

MEMORY_CONSTRAINT_VIOLATION

The dynamic memory range does not satisfy the following constraint.

Signature:

```
MEMORY_CONSTRAINT_VIOLATION(constraint)
<!--NeedCopy-->
```

MEMORY_CONSTRAINT_VIOLATION_MAXPIN

The dynamic memory range violates constraint static_min = dynamic_min = dynamic_max = static_max.

Signature:

```
MEMORY_CONSTRAINT_VIOLATION_MAXPIN(reason)
<!--NeedCopy-->
```

MEMORY_CONSTRAINT_VIOLATION_ORDER

The dynamic memory range violates constraint static_min <= dynamic_min <= dynamic_max <= static_max.

No parameters.

MESSAGE_DEPRECATED

This message has been deprecated.

No parameters.

MESSAGE_METHOD_UNKNOWN

You tried to call a method that does not exist. The method name that you used is echoed.

Signature:

```
MESSAGE_METHOD_UNKNOWN(method)
<!--NeedCopy-->
```

MESSAGE_PARAMETER_COUNT_MISMATCH

You tried to call a method with the incorrect number of parameters. The fully-qualified method name that you used, and the number of received and expected parameters are returned.

Signature:

```
MESSAGE_PARAMETER_COUNT_MISMATCH(method, expected, received)
<!--NeedCopy-->
```

MESSAGE_REMOVED

This function is no longer available.

No parameters.

MIRROR_FAILED

The VDI mirroring cannot be performed

Signature:

```
MIRROR_FAILED(vdi)
<!--NeedCopy-->
```

MISSING_CONNECTION_DETAILS

The license-server connection details (address or port) were missing or incomplete.

No parameters.

MULTIPLE_UPDATE_REPOSITORIES_ENABLED

There is more than one update repository being enabled.

No parameters.

NETWORK_ALREADY_CONNECTED

You tried to create a PIF, but the network you tried to attach it to is already attached to some other PIF, and so the creation failed.

Signature:

```
NETWORK_ALREADY_CONNECTED(network, connected PIF)
<!--NeedCopy-->
```

NETWORK_CONTAINS_PIF

The network contains active PIFs and cannot be deleted.

Signature:

```
NETWORK_CONTAINS_PIF(pifs)
<!--NeedCopy-->
```

NETWORK_CONTAINS_VIF

The network contains active VIFs and cannot be deleted.

Signature:

```
NETWORK_CONTAINS_VIF(vifs)
<!--NeedCopy-->
```

NETWORK_HAS_INCOMPATIBLE_SRIOV_PIFS

The PIF is not compatible with the selected SR-IOV network

Signature:

```
NETWORK_HAS_INCOMPATIBLE_SRIOV_PIFS(PIF, network)
<!--NeedCopy-->
```

NETWORK_HAS_INCOMPATIBLE_VLAN_ON_SRIOV_PIFS

VLAN on the PIF is not compatible with the selected SR-IOV VLAN network

Signature:

```
NETWORK_HAS_INCOMPATIBLE_VLAN_ON_SRIOV_PIFS(PIF, network)
<!--NeedCopy-->
```

NETWORK_INCOMPATIBLE_PURPOSES

You tried to add a purpose to a network but the new purpose is not compatible with an existing purpose of the network or other networks.

Signature:

```
NETWORK_INCOMPATIBLE_PURPOSES(new_purpose, conflicting_purpose)
<!--NeedCopy-->
```

NETWORK_INCOMPATIBLE_WITH_BOND

The network is incompatible with bond

Signature:

```
NETWORK_INCOMPATIBLE_WITH_BOND(network)
<!--NeedCopy-->
```

NETWORK_INCOMPATIBLE_WITH_SRIOV

The network is incompatible with sriov

Signature:

```
NETWORK_INCOMPATIBLE_WITH_SRIOV(network)
<!--NeedCopy-->
```

NETWORK_INCOMPATIBLE_WITH_TUNNEL

The network is incompatible with tunnel

Signature:

```
NETWORK_INCOMPATIBLE_WITH_TUNNEL(network)
<!--NeedCopy-->
```

NETWORK_INCOMPATIBLE_WITH_VLAN_ON_BRIDGE

The network is incompatible with vlan on bridge

Signature:

```
NETWORK_INCOMPATIBLE_WITH_VLAN_ON_BRIDGE(network)
<!--NeedCopy-->
```

NETWORK_INCOMPATIBLE_WITH_VLAN_ON_SRIOV

The network is incompatible with vlan on sriov

Signature:

```
NETWORK_INCOMPATIBLE_WITH_VLAN_ON_SRIOV(network)
<!--NeedCopy-->
```

NETWORK_SRIOV_ALREADY_ENABLED

The PIF selected for the SR-IOV network is already enabled

Signature:

```
NETWORK_SRIOV_ALREADY_ENABLED(PIF)
<!--NeedCopy-->
```

NETWORK_SRIOV_DISABLE_FAILED

Failed to disable SR-IOV on PIF

Signature:

```
NETWORK_SRIOV_DISABLE_FAILED(PIF, msg)
<!--NeedCopy-->
```

NETWORK_SRIOV_ENABLE_FAILED

Failed to enable SR-IOV on PIF

Signature:

```
NETWORK_SRIOV_ENABLE_FAILED(PIF, msg)
<!--NeedCopy-->
```

NETWORK_SRIOV_INSUFFICIENT_CAPACITY

There is insufficient capacity for VF reservation

Signature:

```
NETWORK_SRIOV_INSUFFICIENT_CAPACITY(network)
<!--NeedCopy-->
```

NETWORK_UNMANAGED

The network is not managed by xapi.

Signature:

```
NETWORK_UNMANAGED(network)
<!--NeedCopy-->
```

NOT_ALLOWED_ON_OEM_EDITION

This command is not allowed on the OEM edition.

Signature:

```
NOT_ALLOWED_ON_OEM_EDITION(command)
<!--NeedCopy-->
```

NOT_IMPLEMENTED

The function is not implemented

Signature:

```
NOT_IMPLEMENTED(function)
<!--NeedCopy-->
```

NOT_IN_EMERGENCY_MODE

This pool is not in emergency mode.

No parameters.

NOT_SUPPORTED_DURING_UPGRADE

This operation is not supported during an upgrade.

No parameters.

NOT_SYSTEM_DOMAIN

The given VM is not registered as a system domain. This operation can only be performed on a registered system domain.

Signature:

```
NOT_SYSTEM_DOMAIN(vm)
<!--NeedCopy-->
```

NO_CLUSTER_HOSTS_REACHABLE

No other cluster host was reachable when joining

Signature:

```
NO_CLUSTER_HOSTS_REACHABLE(cluster)
<!--NeedCopy-->
```

NO_COMPATIBLE_CLUSTER_HOST

Clustering is not enabled on this host or pool.

Signature:

```
NO_COMPATIBLE_CLUSTER_HOST(host)
<!--NeedCopy-->
```

NO_HOSTS_AVAILABLE

There were no servers available to complete the specified operation.

No parameters.

NO_MORE_REDO_LOGS_ALLOWED

The upper limit of active redo log instances was reached.

No parameters.

NO_REPOSITORIES_CONFIGURED

No update repositories have been configured.

No parameters.

NO_REPOSITORY_ENABLED

There is no repository being enabled.

No parameters.

NVIDIA_SRIOV_MISCONFIGURED

The NVidia GPU is not configured for SR-IOV as expected

Signature:

```
NVIDIA_SRIOV_MISCONFIGURED(host, device_name)
<!--NeedCopy-->
```

NVIDIA_TOOLS_ERROR

Nvidia tools error. Please ensure that the latest Nvidia tools are installed

Signature:

```
NVIDIA_TOOLS_ERROR(host)
<!--NeedCopy-->
```

OBJECT_NOLONGER_EXISTS

The specified object no longer exists.

No parameters.

ONLY_ALLOWED_ON_OEM_EDITION

This command is only allowed on the OEM edition.

Signature:

```
ONLY_ALLOWED_ON_OEM_EDITION(command)
<!--NeedCopy-->
```

OPENVSWITCH_NOT_ACTIVE

This operation needs the OpenVSwitch networking backend to be enabled on all hosts in the pool.

No parameters.

OPERATION_BLOCKED

You attempted an operation that was explicitly blocked (see the blocked_operations field of the given object).

Signature:

```
OPERATION_BLOCKED(ref, code)
<!--NeedCopy-->
```

OPERATION_NOT_ALLOWED

You attempted an operation that was not allowed.

Signature:

```
OPERATION_NOT_ALLOWED(reason)
<!--NeedCopy-->
```

OPERATION_PARTIALLY_FAILED

Some VMs belonging to the appliance threw an exception while carrying out the specified operation

Signature:

```
OPERATION_PARTIALLY_FAILED(operation)
<!--NeedCopy-->
```

OTHER_OPERATION_IN_PROGRESS

Another operation involving the object is currently in progress

Signature:

```
OTHER_OPERATION_IN_PROGRESS(class, object)
<!--NeedCopy-->
```

OUT_OF_SPACE

There is not enough space to upload the update

Signature:

```
OUT_OF_SPACE(location)
<!--NeedCopy-->
```

PATCH_ALREADY_APPLIED

This patch has already been applied

Signature:

```
PATCH_ALREADY_APPLIED(patch)
<!--NeedCopy-->
```

PATCH_ALREADY_EXISTS

The uploaded patch file already exists

Signature:

```
PATCH_ALREADY_EXISTS(uuid)
<!--NeedCopy-->
```

PATCH_APPLY_FAILED

The patch apply failed. Please see attached output.

Signature:

```
PATCH_APPLY_FAILED(output)
<!--NeedCopy-->
```

PATCH_APPLY_FAILED_BACKUP_FILES_EXIST

The patch apply failed: there are backup files created while applying patch. Please remove these backup files before applying patch again.

Signature:

```
PATCH_APPLY_FAILED_BACKUP_FILES_EXIST(output)
<!--NeedCopy-->
```

PATCH_IS_APPLIED

The specified patch is applied and cannot be destroyed.

No parameters.

PATCH_PRECHECK_FAILED_ISO_MOUNTED

Tools ISO must be ejected from all running VMs.

Signature:

```
PATCH_PRECHECK_FAILED_ISO_MOUNTED(patch)
<!--NeedCopy-->
```

PATCH_PRECHECK_FAILED_OUT_OF_SPACE

The patch pre-check stage failed: the server does not have enough space.

Signature:

```
PATCH_PRECHECK_FAILED_OUT_OF_SPACE(patch, found_space,
    required_required)
<!--NeedCopy-->
```

PATCH_PRECHECK_FAILED_PREREQUISITE_MISSING

The patch pre-check stage failed: prerequisite patches are missing.

Signature:

```
PATCH_PRECHECK_FAILED_PREREQUISITE_MISSING(patch,
    prerequisite_patch_uuid_list)
<!--NeedCopy-->
```

PATCH_PRECHECK_FAILED_UNKNOWN_ERROR

The patch pre-check stage failed with an unknown error. See attached info for more details.

Signature:

```
PATCH_PRECHECK_FAILED_UNKNOWN_ERROR(patch, info)
<!--NeedCopy-->
```

PATCH_PRECHECK_FAILED_VM_RUNNING

The patch pre-check stage failed: there are one or more VMs still running on the server. All VMs must be suspended before the patch can be applied.

Signature:

```
PATCH_PRECHECK_FAILED_VM_RUNNING(patch)
<!--NeedCopy-->
```

PATCH_PRECHECK_FAILED_WRONG_SERVER_BUILD

The patch pre-check stage failed: the server is of an incorrect build.

Signature:

```
PATCH_PRECHECK_FAILED_WRONG_SERVER_BUILD(patch, found_build,
    required_build)
<!--NeedCopy-->
```

PATCH_PRECHECK_FAILED_WRONG_SERVER_VERSION

The patch pre-check stage failed: the server is of an incorrect version.

Signature:

```
PATCH_PRECHECK_FAILED_WRONG_SERVER_VERSION(patch, found_version,
    required_version)
<!--NeedCopy-->
```

PBD_EXISTS

A PBD already exists connecting the SR to the server.

Signature:

```
PBD_EXISTS(sr, host, pbd)
<!--NeedCopy-->
```

PERMISSION_DENIED

Caller not allowed to perform this operation.

Signature:

```
PERMISSION_DENIED(message)
<!--NeedCopy-->
```

PGPU_INSUFFICIENT_CAPACITY_FOR_VGPU

There is insufficient capacity on this PGPU to run the VGPU.

Signature:

```
PGPU_INSUFFICIENT_CAPACITY_FOR_VGPU(pgpu, vgpu_type)
<!--NeedCopy-->
```

PGPU_IN_USE_BY_VM

This PGPU is currently in use by running VMs.

Signature:

```
PGPU_IN_USE_BY_VM(VMs)
<!--NeedCopy-->
```

PGPU_NOT_COMPATIBLE_WITH_GPU_GROUP

PGPU type not compatible with destination group.

Signature:

```
PGPU_NOT_COMPATIBLE_WITH_GPU_GROUP(type, group_types)
<!--NeedCopy-->
```

PIF_ALLOWS_UNPLUG

The operation you requested cannot be performed because the specified PIF allows unplug.

Signature:

```
PIF_ALLOWS_UNPLUG(PIF)
<!--NeedCopy-->
```

PIF_ALREADY_BONDED

This operation cannot be performed because the pif is bonded.

Signature:

```
PIF_ALREADY_BONDED(PIF)
<!--NeedCopy-->
```

PIF_BOND_MORE_THAN_ONE_IP

Only one PIF on a bond is allowed to have an IP configuration.

No parameters.

PIF_BOND_NEEDS_MORE_MEMBERS

A bond must consist of at least two member interfaces

No parameters.

PIF_CANNOT_BOND_CROSS_HOST

You cannot bond interfaces across different servers.

No parameters.

PIF_CONFIGURATION_ERROR

An unknown error occurred while attempting to configure an interface.

Signature:

```
PIF_CONFIGURATION_ERROR(PIF, msg)
<!--NeedCopy-->
```

PIF_DEVICE_NOT_FOUND

The specified device was not found.

No parameters.

PIF_DOES_NOT_ALLOW_UNPLUG

The operation you requested cannot be performed because the specified PIF does not allow unplug.

Signature:

```
PIF_DOES_NOT_ALLOW_UNPLUG(PIF)
<!--NeedCopy-->
```

PIF_HAS_FCOE_SR_IN_USE

The operation you requested cannot be performed because the specified PIF has FCoE SR in use.

Signature:

```
PIF_HAS_FCOE_SR_IN_USE(PIF, SR)
<!--NeedCopy-->
```

PIF_HAS_NO_NETWORK_CONFIGURATION

PIF has no IP configuration (mode currently set to 'none')

Signature:

```
PIF_HAS_NO_NETWORK_CONFIGURATION(PIF)
<!--NeedCopy-->
```

PIF_HAS_NO_V6_NETWORK_CONFIGURATION

PIF has no IPv6 configuration (mode currently set to 'none')

Signature:

```
PIF_HAS_NO_V6_NETWORK_CONFIGURATION(PIF)
<!--NeedCopy-->
```

PIF_INCOMPATIBLE_PRIMARY_ADDRESS_TYPE

The primary address types are not compatible

Signature:

```
PIF_INCOMPATIBLE_PRIMARY_ADDRESS_TYPE(PIF)
<!--NeedCopy-->
```

PIF_IS_MANAGEMENT_INTERFACE

The operation you requested cannot be performed because the specified PIF is the management interface.

Signature:

```
PIF_IS_MANAGEMENT_INTERFACE(PIF)
<!--NeedCopy-->
```

PIF_IS_NOT_PHYSICAL

You tried to perform an operation which is only available on physical PIF

Signature:

```
PIF_IS_NOT_PHYSICAL(PIF)
<!--NeedCopy-->
```

PIF_IS_NOT_SRIOV_CAPABLE

The selected PIF is not capable of network SR-IOV

Signature:

```
PIF_IS_NOT_SRIOV_CAPABLE(PIF)
<!--NeedCopy-->
```

PIF_IS_PHYSICAL

You tried to destroy a PIF, but it represents an aspect of the physical host configuration, and so cannot be destroyed. The parameter echoes the PIF handle you gave.

Signature:

```
PIF_IS_PHYSICAL(PIF)
<!--NeedCopy-->
```

PIF_IS_SRIOV_LOGICAL

You tried to create a bond on top of a network SR-IOV logical PIF - use the underlying physical PIF instead

Signature:

```
PIF_IS_SRIOV_LOGICAL(PIF)
<!--NeedCopy-->
```

PIF_IS_VLAN

You tried to create a VLAN on top of another VLAN - use the underlying physical PIF/bond instead

Signature:

```
PIF_IS_VLAN(PIF)
<!--NeedCopy-->
```

PIF_NOT_ATTACHED_TO_HOST

Cluster_host creation failed as the PIF provided is not attached to the host.

Signature:

```
PIF_NOT_ATTACHED_TO_HOST(pif, host)
<!--NeedCopy-->
```

PIF_NOT_PRESENT

This host has no PIF on the given network.

Signature:

```
PIF_NOT_PRESENT(host, network)
<!--NeedCopy-->
```

PIF_SRIOV_STILL_EXISTS

The PIF is still related with a network SR-IOV

Signature:

```
PIF_SRIOV_STILL_EXISTS(PIF)
<!--NeedCopy-->
```

PIF_TUNNEL_STILL_EXISTS

Operation cannot proceed while a tunnel exists on this interface.

Signature:

```
PIF_TUNNEL_STILL_EXISTS(PIF)
<!--NeedCopy-->
```

PIF_UNMANAGED

The operation you requested cannot be performed because the specified PIF is not managed by xapi.

Signature:

```
PIF_UNMANAGED(PIF)
<!--NeedCopy-->
```

PIF_VLAN_EXISTS

You tried to create a PIF, but it already exists.

Signature:

```
PIF_VLAN_EXISTS(PIF)
<!--NeedCopy-->
```

PIF_VLAN_STILL_EXISTS

Operation cannot proceed while a VLAN exists on this interface.

Signature:

```
PIF_VLAN_STILL_EXISTS(PIF)
<!--NeedCopy-->
```

POOL_AUTH_ALREADY_ENABLED

External authentication is already enabled for at least one server in this pool.

Signature:

```
POOL_AUTH_ALREADY_ENABLED(host)
<!--NeedCopy-->
```

POOL_AUTH_DISABLE_FAILED

The pool failed to disable the external authentication of at least one host.

Signature:

```
POOL_AUTH_DISABLE_FAILED(host, message)
<!--NeedCopy-->
```

POOL_AUTH_DISABLE_FAILED_INVALID_ACCOUNT

External authentication has been disabled with errors: Some AD machine accounts were not disabled on the AD server due to invalid account.

Signature:

```
POOL_AUTH_DISABLE_FAILED_INVALID_ACCOUNT(host, message)
<!--NeedCopy-->
```

POOL_AUTH_DISABLE_FAILED_PERMISSION_DENIED

External authentication has been disabled with errors: Your AD machine account was not disabled on the AD server as permission was denied.

Signature:

```
POOL_AUTH_DISABLE_FAILED_PERMISSION_DENIED(host, message)
<!--NeedCopy-->
```

POOL_AUTH_DISABLE_FAILED_WRONG_CREDENTIALS

External authentication has been disabled with errors: Some AD machine accounts were not disabled on the AD server due to invalid credentials.

Signature:

```
POOL_AUTH_DISABLE_FAILED_WRONG_CREDENTIALS(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_DOMAIN_LOOKUP_FAILED

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_DOMAIN_LOOKUP_FAILED(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_DUPLICATE_HOSTNAME

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_DUPLICATE_HOSTNAME(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_INVALID_ACCOUNT

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_INVALID_ACCOUNT(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_INVALID_OU

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_INVALID_OU(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_PERMISSION_DENIED

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_PERMISSION_DENIED(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_UNAVAILABLE

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_UNAVAILABLE(host, message)
<!--NeedCopy-->
```

POOL_AUTH_ENABLE_FAILED_WRONG_CREDENTIALS

The pool failed to enable external authentication.

Signature:

```
POOL_AUTH_ENABLE_FAILED_WRONG_CREDENTIALS(host, message)
<!--NeedCopy-->
```

POOL_JOINING_EXTERNAL_AUTH_MISMATCH

Cannot join pool whose external authentication configuration is different.

No parameters.

POOL_JOINING_HOST_CA_CERTIFICATES_CONFLICT

The host joining the pool has different CA certificates from the pool coordinator while using the same name, uninstall them and try again.

No parameters.

POOL_JOINING_HOST_HAS_BONDS

The host joining the pool must not have any bonds.

No parameters.

POOL_JOINING_HOST_HAS_NETWORK_SRIOVS

The host joining the pool must not have any network SR-IOVs.

No parameters.

POOL_JOINING_HOST_HAS_NON_MANAGEMENT_VLANS

The host joining the pool must not have any non-management vlans.

No parameters.

POOL_JOINING_HOST_HAS_TUNNELS

The host joining the pool must not have any tunnels.

No parameters.

POOL_JOINING_HOST_MANAGEMENT_VLAN_DOES_NOT_MATCH

The host joining the pool must have the same management vlan.

Signature:

```
POOL_JOINING_HOST_MANAGEMENT_VLAN_DOES_NOT_MATCH(local, remote)
<!--NeedCopy-->
```

POOL_JOINING_HOST_MUST_HAVE_PHYSICAL_MANAGEMENT_NIC

The server joining the pool must have a physical management NIC (i.e. the management NIC must not be on a VLAN or bonded PIF).

No parameters.

POOL_JOINING_HOST_MUST_HAVE_SAME_API_VERSION

The host joining the pool must have the same API version as the pool coordinator.

Signature:

```
POOL_JOINING_HOST_MUST_HAVE_SAME_API_VERSION(host_api_version,
    master_api_version)
<!--NeedCopy-->
```

POOL_JOINING_HOST_MUST_HAVE_SAME_DB_SCHEMA

The host joining the pool must have the same database schema as the pool coordinator.

Signature:

```
POOL_JOINING_HOST_MUST_HAVE_SAME_DB_SCHEMA(host_db_schema,
    master_db_schema)
<!--NeedCopy-->
```

POOL_JOINING_HOST_MUST_HAVE_SAME_PRODUCT_VERSION

The server joining the pool must have the same product version as the pool coordinator.

No parameters.

POOL_JOINING_HOST_MUST_ONLY_HAVE_PHYSICAL_PIFS

The host joining the pool must not have any bonds, VLANs or tunnels.

No parameters.

PROVISION_FAILED_OUT_OF_SPACE

The provision call failed because it ran out of space.

No parameters.

PROVISION_ONLY_ALLOWED_ON_TEMPLATE

The provision call can only be invoked on templates, not regular VMs.

No parameters.

PUSB_VDI_CONFLICT

The VDI corresponding to this PUSB has existing VBDs.

Signature:

```
PUSB_VDI_CONFLICT(PUSB, VDI)
<!--NeedCopy-->
```

PVS_CACHE_STORAGE_ALREADY_PRESENT

The PVS site already has cache storage configured for the host.

Signature:

```
PVS_CACHE_STORAGE_ALREADY_PRESENT(site, host)
<!--NeedCopy-->
```

PVS_CACHE_STORAGE_IS_IN_USE

The PVS cache storage is in use by the site and cannot be removed.

Signature:

```
PVS_CACHE_STORAGE_IS_IN_USE(PVS_cache_storage)
<!--NeedCopy-->
```

PVS_PROXY_ALREADY_PRESENT

The VIF is already associated with a PVS proxy

Signature:

```
PVS_PROXY_ALREADY_PRESENT(proxies)
<!--NeedCopy-->
```

PVS_SERVER_ADDRESS_IN_USE

The address specified is already in use by an existing PVS_server object

Signature:

```
PVS_SERVER_ADDRESS_IN_USE(address)
<!--NeedCopy-->
```

PVS_SITE_CONTAINS_RUNNING_PROXIES

The PVS site contains running proxies.

Signature:

```
PVS_SITE_CONTAINS_RUNNING_PROXIES(proxies)
<!--NeedCopy-->
```

PVS_SITE_CONTAINS_SERVERS

The PVS site contains servers and cannot be forgotten.

Signature:

```
PVS_SITE_CONTAINS_SERVERS(servers)
<!--NeedCopy-->
```

RBAC_PERMISSION_DENIED

RBAC permission denied.

Signature:

```
RBAC_PERMISSION_DENIED(permission, message)
<!--NeedCopy-->
```

REDO_LOG_IS_ENABLED

The operation could not be performed because a redo log is enabled on the Pool.

No parameters.

REPOSITORY_ALREADY_EXISTS

The repository already exists.

Signature:

```
REPOSITORY_ALREADY_EXISTS(ref)
<!--NeedCopy-->
```

REPOSITORY_CLEANUP_FAILED

Failed to clean up local repository on coordinator.

No parameters.

REPOSITORY_IS_IN_USE

The repository is in use.

No parameters.

REPOSYNC_FAILED

Syncing with remote YUM repository failed.

No parameters.

REQUIRED_PIF_IS_UNPLUGGED

The operation you requested cannot be performed because the specified PIF is currently unplugged.

Signature:

```
REQUIRED_PIF_IS_UNPLUGGED(PIF)
<!--NeedCopy-->
```

RESTORE_INCOMPATIBLE_VERSION

The restore could not be performed because this backup has been created by a different (incompatible) product version

No parameters.

RESTORE_SCRIPT_FAILED

The restore could not be performed because the restore script failed. Is the file corrupt?

Signature:

```
RESTORE_SCRIPT_FAILED(log)
<!--NeedCopy-->
```

RESTORE_TARGET_MGMT_IF_NOT_IN_BACKUP

The restore could not be performed because the server's current management interface is not in the backup. The interfaces mentioned in the backup are:

No parameters.

RESTORE_TARGET_MISSING_DEVICE

The restore could not be performed because a network interface is missing

Signature:

```
RESTORE_TARGET_MISSING_DEVICE(device)
<!--NeedCopy-->
```

ROLE_ALREADY_EXISTS

Role already exists.

No parameters.

ROLE_NOT_FOUND

Role cannot be found.

No parameters.

SERVER_CERTIFICATE_CHAIN_INVALID

The provided intermediate certificates are not in a PEM-encoded X509.

No parameters.

SERVER_CERTIFICATE_EXPIRED

The provided certificate has expired.

Signature:

```
SERVER_CERTIFICATE_EXPIRED(now, not_after)
<!--NeedCopy-->
```

SERVER_CERTIFICATE_INVALID

The provided certificate is not in a PEM-encoded X509.

No parameters.

SERVER_CERTIFICATE_KEY_ALGORITHM_NOT_SUPPORTED

The provided key uses an unsupported algorithm.

Signature:

```
SERVER_CERTIFICATE_KEY_ALGORITHM_NOT_SUPPORTED(algorithm_oid)
<!--NeedCopy-->
```

SERVER_CERTIFICATE_KEY_INVALID

The provided key is not in a PEM-encoded PKCS#8 format.

No parameters.

SERVER_CERTIFICATE_KEY_MISMATCH

The provided key does not match the provided certificate's public key.

No parameters.

SERVER_CERTIFICATE_KEY_RSA_LENGTH_NOT_SUPPORTED

The provided RSA key does not have a length between 2048 and 4096.

Signature:

```
SERVER_CERTIFICATE_KEY_RSA_LENGTH_NOT_SUPPORTED(length)
<!--NeedCopy-->
```

SERVER_CERTIFICATE_KEY_RSA_MULTI_NOT_SUPPORTED

The provided RSA key is using more than 2 primes, expecting only 2.

No parameters.

SERVER_CERTIFICATE_NOT_VALID_YET

The provided certificate is not valid yet.

Signature:

```
SERVER_CERTIFICATE_NOT_VALID_YET(now, not_before)
<!--NeedCopy-->
```

SERVER_CERTIFICATE_SIGNATURE_NOT_SUPPORTED

The provided certificate is not using the SHA256 (SHA2) signature algorithm.

No parameters.

SESSION_AUTHENTICATION_FAILED

The credentials given by the user are incorrect, so access has been denied, and you have not been issued a session handle.

No parameters.

SESSION_AUTHORIZATION_FAILED

The credentials given by the user are correct, but the user could not be authorized, so access has been denied, and you have not been issued a session handle.

Signature:

```
SESSION_AUTHORIZATION_FAILED(username, msg)
<!--NeedCopy-->
```

SESSION_INVALID

You gave an invalid session reference. It may have been invalidated by a server restart, or timed out. You should get a new session handle, using one of the session.login_ calls. This error does not invalidate the current connection. The handle parameter echoes the bad value given.

Signature:

```
SESSION_INVALID(handle)
<!--NeedCopy-->
```

SESSION_NOT_REGISTERED

This session is not registered to receive events. You must call event.register before event.next. The session handle you are using is echoed.

Signature:

```
SESSION_NOT_REGISTERED(handle)
<!--NeedCopy-->
```

SLAVE_REQUIRES_MANAGEMENT_INTERFACE

The management interface on a supporter cannot be disabled because the supporter would enter emergency mode.

No parameters.

SM_PLUGIN_COMMUNICATION_FAILURE

The SM plug-in did not respond to a query.

Signature:

```
SM_PLUGIN_COMMUNICATION_FAILURE(sm)
<!--NeedCopy-->
```

SR_ATTACH_FAILED

Attaching this SR failed.

Signature:

```
SR_ATTACH_FAILED(sr)
<!--NeedCopy-->
```

SR_BACKEND_FAILURE

There was an SR backend failure.

Signature:

```
SR_BACKEND_FAILURE(status, stdout, stderr)
<!--NeedCopy-->
```

SR_DEVICE_IN_USE

The SR operation cannot be performed because a device underlying the SR is in use by the server.

No parameters.

SR_DOES_NOT_SUPPORT_MIGRATION

Cannot migrate a VDI to or from an SR that doesn't support migration.

Signature:

```
SR_DOES_NOT_SUPPORT_MIGRATION(sr)
<!--NeedCopy-->
```

SR_FULL

The SR is full. Requested new size exceeds the maximum size

Signature:

```
SR_FULL(requested, maximum)
<!--NeedCopy-->
```

SR_HAS_MULTIPLE_PBDS

The SR.shared flag cannot be set to false while the SR remains connected to multiple servers.

Signature:

```
SR_HAS_MULTIPLE_PBDS(PBD)
<!--NeedCopy-->
```

SR_HAS_NO_PBDS

The SR has no attached PBDs

Signature:

```
SR_HAS_NO_PBDS(sr)
<!--NeedCopy-->
```

SR_HAS_PBD

The SR is still connected to a host via a PBD. It cannot be destroyed or forgotten.

Signature:

```
SR_HAS_PBD(sr)
<!--NeedCopy-->
```

SR_INDESTRUCTIBLE

The SR could not be destroyed because the 'indestructible'flag was set on it.

Signature:

```
SR_INDESTRUCTIBLE(sr)
<!--NeedCopy-->
```

SR_IS_CACHE_SR

The SR is currently being used as a local cache SR.

Signature:

```
SR_IS_CACHE_SR(host)
<!--NeedCopy-->
```

SR_NOT_ATTACHED

The SR is not attached.

Signature:

```
SR_NOT_ATTACHED(sr)
<!--NeedCopy-->
```

SR_NOT_EMPTY

The SR operation cannot be performed because the SR is not empty.

No parameters.

SR_NOT_SHARABLE

The PBD could not be plugged because the SR is in use by another host and is not marked as sharable.

Signature:

```
SR_NOT_SHARABLE(sr, host)
<!--NeedCopy-->
```

SR_OPERATION_NOT_SUPPORTED

The SR backend does not support the operation (check the SR's allowed operations)

Signature:

```
SR_OPERATION_NOT_SUPPORTED(sr)
<!--NeedCopy-->
```

SR_REQUIRES_UPGRADE

The operation cannot be performed until the SR has been upgraded

Signature:

```
SR_REQUIRES_UPGRADE(SR)
<!--NeedCopy-->
```

SR_SOURCE_SPACE_INSUFFICIENT

The source SR does not have sufficient temporary space available to proceed the operation.

Signature:

```
SR_SOURCE_SPACE_INSUFFICIENT(sr)
<!--NeedCopy-->
```

SR_UNKNOWN_DRIVER

The SR could not be connected because the driver was not recognised.

Signature:

```
SR_UNKNOWN_DRIVER(driver)
<!--NeedCopy-->
```

SR_UUID_EXISTS

An SR with that uuid already exists.

Signature:

```
SR_UUID_EXISTS(uuid)
<!--NeedCopy-->
```

SR_VDI_LOCKING_FAILED

The operation could not proceed because necessary VDIs were already locked at the storage level.

No parameters.

SSL_VERIFY_ERROR

The remote system's SSL certificate failed to verify against our certificate library.

Signature:

```
SSL_VERIFY_ERROR(reason)
<!--NeedCopy-->
```

SUBJECT_ALREADY_EXISTS

Subject already exists.

No parameters.

SUBJECT_CANNOT_BE_RESOLVED

Subject cannot be resolved by the external directory service.

No parameters.

SUSPEND_IMAGE_NOT_ACCESSIBLE

The suspend image of a checkpoint is not accessible from the host on which the VM is running

Signature:

```
SUSPEND_IMAGE_NOT_ACCESSIBLE(vdi)
<!--NeedCopy-->
```

SYNC_UPDATES_IN_PROGRESS

The operation could not be performed because syncing updates is in progress.

No parameters.

SYSTEM_STATUS_MUST_USE_TAR_ON_OEM

You must use tar output to retrieve system status from an OEM server.

No parameters.

SYSTEM_STATUS_RETRIEVAL_FAILED

Retrieving system status from the host failed. A diagnostic reason suitable for support organisations is also returned.

Signature:

```
SYSTEM_STATUS_RETRIEVAL_FAILED(reason)
<!--NeedCopy-->
```

TASK_CANCELLED

The request was asynchronously canceled.

Signature:

```
TASK_CANCELLED(task)
<!--NeedCopy-->
```

TELEMETRY_NEXT_COLLECTION_TOO_LATE

The next scheduled telemetry data collection is too far into the future. Pick a timestamp within two telemetry intervals starting from now.

Signature:

```
TELEMETRY_NEXT_COLLECTION_TOO_LATE(timestamp)
<!--NeedCopy-->
```

TLS_CONNECTION_FAILED

Cannot contact the other host using TLS on the specified address and port

Signature:

```
TLS_CONNECTION_FAILED(address, port)
<!--NeedCopy-->
```

TOO_BUSY

The request was rejected because the server is too busy.

No parameters.

TOO_MANY_PENDING_TASKS

The request was rejected because there are too many pending tasks on the server.

No parameters.

TOO_MANY_STORAGE_MIGRATES

You reached the maximal number of concurrently migrating VMs.

Signature:

```
TOO_MANY_STORAGE_MIGRATES(number)
<!--NeedCopy-->
```

TOO_MANY_VUSBS

The VM has too many VUSBs.

Signature:

```
TOO_MANY_VUSBS(number)
<!--NeedCopy-->
```

TRANSPORT_PIF_NOT_CONFIGURED

The tunnel transport PIF has no IP configuration set.

Signature:

```
TRANSPORT_PIF_NOT_CONFIGURED(PIF)
<!--NeedCopy-->
```

UNIMPLEMENTED_IN_SM_BACKEND

You have attempted a function which is not implemented

Signature:

```
UNIMPLEMENTED_IN_SM_BACKEND(message)
<!--NeedCopy-->
```

UNKNOWN_BOOTLOADER

The requested bootloader is unknown

Signature:

```
UNKNOWN_BOOTLOADER(vm, bootloader)
<!--NeedCopy-->
```

UPDATEINFO_HASH_MISMATCH

The hash of updateinfo doesn't match with current one. There may be newer available updates.

No parameters.

UPDATES_REQUIRE_RECOMMENDED_GUIDANCE

Requires recommended guidance after applying updates.

Signature:

```
UPDATES_REQUIRE_RECOMMENDED_GUIDANCE(recommended_guidance)
<!--NeedCopy-->
```

UPDATE_ALREADY_APPLIED

This update has already been applied.

Signature:

```
UPDATE_ALREADY_APPLIED(update)
<!--NeedCopy-->
```

UPDATE_ALREADY_APPLIED_IN_POOL

This update has already been applied to all hosts in the pool.

Signature:

```
UPDATE_ALREADY_APPLIED_IN_POOL(update)
<!--NeedCopy-->
```

UPDATE_ALREADY_EXISTS

The uploaded update already exists

Signature:

```
UPDATE_ALREADY_EXISTS(uuid)
<!--NeedCopy-->
```

UPDATE_APPLY_FAILED

The update failed to apply. Please see attached output.

Signature:

```
UPDATE_APPLY_FAILED(output)
<!--NeedCopy-->
```

UPDATE_GUIDANCE_CHANGED

Guidance for the update has changed

Signature:

```
UPDATE_GUIDANCE_CHANGED(guidance)
<!--NeedCopy-->
```

UPDATE_IS_APPLIED

The specified update has been applied and cannot be destroyed.

No parameters.

UPDATE_POOL_APPLY_FAILED

The update cannot be applied for the following host(s).

Signature:

```
UPDATE_POOL_APPLY_FAILED(hosts)
<!--NeedCopy-->
```

UPDATE_PRECHECK_FAILED_CONFLICT_PRESENT

The update pre-check stage failed: conflicting update(s) are present.

Signature:

```
UPDATE_PRECHECK_FAILED_CONFLICT_PRESENT(update, conflict_update)
<!--NeedCopy-->
```

UPDATE_PRECHECK_FAILED_GPGKEY_NOT_IMPORTED

The update pre-check stage failed: RPM package validation requires a GPG key that is not present on the host.

Signature:

```
UPDATE_PRECHECK_FAILED_GPGKEY_NOT_IMPORTED(update)
<!--NeedCopy-->
```

UPDATE_PRECHECK_FAILED_OUT_OF_SPACE

The update pre-check stage failed: the server does not have enough space.

Signature:

```
UPDATE_PRECHECK_FAILED_OUT_OF_SPACE(update, available_space,
    required_space )
<!--NeedCopy-->
```

UPDATE_PRECHECK_FAILED_PREREQUISITE_MISSING

The update pre-check stage failed: prerequisite update(s) are missing.

Signature:

```
UPDATE_PRECHECK_FAILED_PREREQUISITE_MISSING(update, prerequisite_update
    )
<!--NeedCopy-->
```

UPDATE_PRECHECK_FAILED_UNKNOWN_ERROR

The update pre-check stage failed with an unknown error.

Signature:

```
UPDATE_PRECHECK_FAILED_UNKNOWN_ERROR(update, info)
<!--NeedCopy-->
```

UPDATE_PRECHECK_FAILED_WRONG_SERVER_VERSION

The update pre-check stage failed: the server is of an incorrect version.

Signature:

```
UPDATE_PRECHECK_FAILED_WRONG_SERVER_VERSION(update, installed_version,
    required_version )
<!--NeedCopy-->
```

USB_ALREADY_ATTACHED

The USB device is currently attached to a VM.

Signature:

```
USB_ALREADY_ATTACHED(PUSB, VM)
<!--NeedCopy-->
```

USB_GROUP_CONFLICT

USB_groups are currently restricted to contain no more than one VUSB.

Signature:

```
USB_GROUP_CONFLICT(USB_group)
<!--NeedCopy-->
```

USB_GROUP_CONTAINS_NO_PUSBS

The USB group does not contain any PUSBs.

Signature:

```
USB_GROUP_CONTAINS_NO_PUSBS(usb_group)
<!--NeedCopy-->
```

USB_GROUP_CONTAINS_PUSB

The USB group contains active PUSBs and cannot be deleted.

Signature:

```
USB_GROUP_CONTAINS_PUSB(pusbs)
<!--NeedCopy-->
```

USB_GROUP_CONTAINS_VUSB

The USB group contains active VUSBs and cannot be deleted.

Signature:

```
USB_GROUP_CONTAINS_VUSB(vusbs)
<!--NeedCopy-->
```

USER_IS_NOT_LOCAL_SUPERUSER

Only the local superuser can perform this operation.

Signature:

```
USER_IS_NOT_LOCAL_SUPERUSER(msg)
<!--NeedCopy-->
```

UUID_INVALID

The uuid you supplied was invalid.

Signature:

```
UUID_INVALID(type, uuid)
<!--NeedCopy-->
```

V6D_FAILURE

There was a problem with the license daemon (v6d).

No parameters.

VALUE_NOT_SUPPORTED

You attempted to set a value that is not supported by this implementation. The fully-qualified field name and the value that you tried to set are returned. Also returned is a developer-only diagnostic reason.

Signature:

```
VALUE_NOT_SUPPORTED(field, value, reason)
<!--NeedCopy-->
```

VBD_CDS_MUST_BE_READONLY

Read/write CDs are not supported

No parameters.

VBD_IS_EMPTY

Operation could not be performed because the drive is empty

Signature:

```
VBD_IS_EMPTY(vbd)
<!--NeedCopy-->
```

VBD_NOT_EMPTY

Operation could not be performed because the drive is not empty

Signature:

```
VBD_NOT_EMPTY(vbd)
<!--NeedCopy-->
```

VBD_NOT_REMOVABLE_MEDIA

Media could not be ejected because it is not removable

Signature:

```
VBD_NOT_REMOVABLE_MEDIA(vbd)
<!--NeedCopy-->
```

VBD_NOT_UNPLUGGABLE

Drive could not be hot-unplugged because it is not marked as unpluggable

Signature:

```
VBD_NOT_UNPLUGGABLE(vbd)
<!--NeedCopy-->
```

VBD_TRAY_LOCKED

This VM has locked the DVD drive tray, so the disk cannot be ejected

Signature:

```
VBD_TRAY_LOCKED(vbd)
<!--NeedCopy-->
```

VCPU_MAX_NOT_CORES_PER_SOCKET_MULTIPLE

VCPUs_max must be a multiple of cores-per-socket

Signature:

```
VCPU_MAX_NOT_CORES_PER_SOCKET_MULTIPLE(vcpu_max, cores_per_socket)
<!--NeedCopy-->
```

VDI_CBT_ENABLED

The requested operation is not allowed for VDIs with CBT enabled or VMs having such VDIs, and CBT is enabled for the specified VDI.

Signature:

```
VDI_CBT_ENABLED(vdi)
<!--NeedCopy-->
```

VDI_CONTAINS_METADATA_OF_THIS_POOL

The VDI could not be opened for metadata recovery as it contains the current pool's metadata.

Signature:

```
VDI_CONTAINS_METADATA_OF_THIS_POOL(vdi, pool)
<!--NeedCopy-->
```

VDI_COPY_FAILED

The VDI copy action has failed

No parameters.

VDI_HAS_RRDS

The operation cannot be performed because this VDI has rrd stats

Signature:

```
VDI_HAS_RRDS(vdi)
<!--NeedCopy-->
```

VDI_INCOMPATIBLE_TYPE

This operation cannot be performed because the specified VDI is of an incompatible type (eg: an HA statefile cannot be attached to a guest)

Signature:

```
VDI_INCOMPATIBLE_TYPE(vdi, type)
<!--NeedCopy-->
```

VDI_IN_USE

This operation cannot be performed because this VDI is in use by some other operation

Signature:

```
VDI_IN_USE(vdi, operation)
<!--NeedCopy-->
```

VDI_IS_A_PHYSICAL_DEVICE

The operation cannot be performed on physical device

Signature:

```
VDI_IS_A_PHYSICAL_DEVICE(vdi)
<!--NeedCopy-->
```

VDI_IS_ENCRYPTED

The requested operation is not allowed because the specified VDI is encrypted.

Signature:

```
VDI_IS_ENCRYPTED(vdi)
<!--NeedCopy-->
```

VDI_IS_NOT_ISO

This operation can only be performed on CD VDIs (iso files or CDROM drives)

Signature:

```
VDI_IS_NOT_ISO(vdi, type)
<!--NeedCopy-->
```

VDI_LOCATION_MISSING

This operation cannot be performed because the specified VDI could not be found in the specified SR

Signature:

```
VDI_LOCATION_MISSING(sr, location)
<!--NeedCopy-->
```

VDI_MISSING

This operation cannot be performed because the specified VDI could not be found on the storage substrate

Signature:

```
VDI_MISSING(sr, vdi)
<!--NeedCopy-->
```

VDI_NEEDS_VM_FOR_MIGRATE

Cannot migrate a VDI which is not attached to a running VM.

Signature:

```
VDI_NEEDS_VM_FOR_MIGRATE(vdi)
<!--NeedCopy-->
```

VDI_NOT_AVAILABLE

This operation cannot be performed because this VDI could not be properly attached to the VM.

Signature:

```
VDI_NOT_AVAILABLE(vdi)
<!--NeedCopy-->
```

VDI_NOT_IN_MAP

This VDI was not mapped to a destination SR in VM.migrate_send operation

Signature:

```
VDI_NOT_IN_MAP(vdi)
<!--NeedCopy-->
```

VDI_NOT_MANAGED

This operation cannot be performed because the system does not manage this VDI

Signature:

```
VDI_NOT_MANAGED(vdi)
<!--NeedCopy-->
```

VDI_NOT_SPARSE

The VDI is not stored using a sparse format. It is not possible to query and manipulate only the changed blocks (or 'block differences'or 'disk deltas') between two VDIs. Please select a VDI which uses a sparse-aware technology such as VHD.

Signature:

```
VDI_NOT_SPARSE(vdi)
<!--NeedCopy-->
```

VDI_NO_CBT_METADATA

The requested operation is not allowed because the specified VDI does not have changed block tracking metadata.

Signature:

```
VDI_NO_CBT_METADATA(vdi)
<!--NeedCopy-->
```

VDI_ON_BOOT_MODE_INCOMPATIBLE_WITH_OPERATION

This operation is not permitted on VDIs in the 'on-boot=reset' mode, or on VMs having such VDIs.

No parameters.

VDI_READONLY

The operation required write access but this VDI is read-only

Signature:

```
VDI_READONLY(vdi)
<!--NeedCopy-->
```

VDI_TOO_LARGE

The VDI is too large.

Signature:

```
VDI_TOO_LARGE(vdi, maximum size)
<!--NeedCopy-->
```

VDI_TOO_SMALL

The VDI is too small. Please resize it to at least the minimum size.

Signature:

```
VDI_TOO_SMALL(vdi, minimum size)
<!--NeedCopy-->
```

VGPU_DESTINATION_INCOMPATIBLE

The VGPU is not compatible with any PGPU in the destination.

Signature:

```
VGPU_DESTINATION_INCOMPATIBLE(reason, vgpu, host)
<!--NeedCopy-->
```

VGPU_GUEST_DRIVER_LIMIT

The guest driver does not support VGPU migration.

Signature:

```
VGPU_GUEST_DRIVER_LIMIT(reason, vm, host)
<!--NeedCopy-->
```

VGPU_SUSPENSION_NOT_SUPPORTED

The VGPU configuration does not support suspension.

Signature:

```
VGPU_SUSPENSION_NOT_SUPPORTED(reason, vgpu, host)
<!--NeedCopy-->
```

VGPU_TYPE_NOT_COMPATIBLE

Cannot create a virtual GPU that is incompatible with the existing types on the VM.

Signature:

```
VGPU_TYPE_NOT_COMPATIBLE(type)
<!--NeedCopy-->
```

VGPU_TYPE_NOT_COMPATIBLE_WITH_RUNNING_TYPE

The VGPU type is incompatible with one or more of the VGPU types currently running on this PGPU

Signature:

```
VGPU_TYPE_NOT_COMPATIBLE_WITH_RUNNING_TYPE(pgpu, type, running_type)
<!--NeedCopy-->
```

VGPU_TYPE_NOT_ENABLED

VGPU type is not one of the PGPU's enabled types.

Signature:

```
VGPU_TYPE_NOT_ENABLED(type, enabled_types)
<!--NeedCopy-->
```

VGPU_TYPE_NOT_SUPPORTED

VGPU type is not one of the PGPU's supported types.

Signature:

```
VGPU_TYPE_NOT_SUPPORTED(type, supported_types)
<!--NeedCopy-->
```

VGPU_TYPE_NO_LONGER_SUPPORTED

VGPU type is no longer supported

Signature:

```
VGPU_TYPE_NO_LONGER_SUPPORTED(type)
<!--NeedCopy-->
```

VIF_IN_USE

Network has active VIFs

Signature:

```
VIF_IN_USE(network, VIF)
<!--NeedCopy-->
```

VIF_NOT_IN_MAP

This VIF was not mapped to a destination Network in VM.migrate_send operation

Signature:

```
VIF_NOT_IN_MAP(vif)
<!--NeedCopy-->
```

VLAN_IN_USE

Operation cannot be performed because this VLAN is already in use. Please check your network configuration.

Signature:

```
VLAN_IN_USE(device, vlan)
<!--NeedCopy-->
```

VLAN_TAG_INVALID

You tried to create a VLAN, but the tag you gave was invalid -- it must be between 0 and 4094. The parameter echoes the VLAN tag you gave.

Signature:

```
VLAN_TAG_INVALID(VLAN)
<!--NeedCopy-->
```

VMPP_ARCHIVE_MORE_FREQUENT_THAN_BACKUP

Archive more frequent than backup.

No parameters.

VMPP_HAS_VM

There is at least one VM assigned to this protection policy.

No parameters.

VMSS_HAS_VM

There is at least one VM assigned to snapshot schedule.

No parameters.

VMS_FAILED_TO_COOPERATE

The given VMs failed to release memory when instructed to do so

No parameters.

VM_ASSIGNED_TO_PROTECTION_POLICY

This VM is assigned to a protection policy.

Signature:

```
VM_ASSIGNED_TO_PROTECTION_POLICY(vm, vmpp)
<!--NeedCopy-->
```

VM_ASSIGNED_TO_SNAPSHOT_SCHEDULE

This VM is assigned to a snapshot schedule.

Signature:

```
VM_ASSIGNED_TO_SNAPSHOT_SCHEDULE(vm, vmss)
<!--NeedCopy-->
```

VM_ATTACHED_TO_MORE_THAN_ONE_VDI_WITH_TIMEOFFSET_MARKED_AS_RESET_ON_BOOT

You attempted to start a VM that's attached to more than one VDI with a timeoffset marked as reset-on-boot.

Signature:

```
VM_ATTACHED_TO_MORE_THAN_ONE_VDI_WITH_TIMEOFFSET_MARKED_AS_RESET_ON_BOOT
    (vm)
<!--NeedCopy-->
```

VM_BAD_POWER_STATE

You attempted an operation on a VM that was not in an appropriate power state at the time; for example, you attempted to start a VM that was already running. The parameters returned are the VM's handle, and the expected and actual VM state at the time of the call.

Signature:

```
VM_BAD_POWER_STATE(vm, expected, actual)
<!--NeedCopy-->
```

VM_BIOS_STRINGS_ALREADY_SET

The BIOS strings for this VM have already been set and cannot be changed.

No parameters.

VM_CALL_PLUGIN_RATE_LIMIT

There is a minimal interval required between consecutive plug-in calls made on the same VM, please wait before retry.

Signature:

```
VM_CALL_PLUGIN_RATE_LIMIT(VM, interval, wait)
<!--NeedCopy-->
```

VM_CANNOT_DELETE_DEFAULT_TEMPLATE

You cannot delete the specified default template.

Signature:

```
VM_CANNOT_DELETE_DEFAULT_TEMPLATE(vm)
<!--NeedCopy-->
```

VM_CHECKPOINT_RESUME_FAILED

An error occured while restoring the memory image of the specified virtual machine

Signature:

```
VM_CHECKPOINT_RESUME_FAILED(vm)
<!--NeedCopy-->
```

VM_CHECKPOINT_SUSPEND_FAILED

An error occured while saving the memory image of the specified virtual machine

Signature:

```
VM_CHECKPOINT_SUSPEND_FAILED(vm)
<!--NeedCopy-->
```

VM_CRASHED

The VM crashed

Signature:

```
VM_CRASHED(vm)
<!--NeedCopy-->
```

VM_DUPLICATE_VBD_DEVICE

The specified VM has a duplicate VBD device and cannot be started.

Signature:

```
VM_DUPLICATE_VBD_DEVICE(vm, vbd, device)
<!--NeedCopy-->
```

VM_FAILED_SHUTDOWN_ACKNOWLEDGMENT

VM didn't acknowledge the need to shutdown.

Signature:

```
VM_FAILED_SHUTDOWN_ACKNOWLEDGMENT(vm)
<!--NeedCopy-->
```

VM_FAILED_SUSPEND_ACKNOWLEDGMENT

VM didn't acknowledge the need to suspend.

Signature:

```
VM_FAILED_SUSPEND_ACKNOWLEDGMENT(vm)
<!--NeedCopy-->
```

VM_HALTED

The VM unexpectedly halted

Signature:

```
VM_HALTED(vm)
<!--NeedCopy-->
```

VM_HAS_CHECKPOINT

Cannot migrate a VM which has a checkpoint.

Signature:

```
VM_HAS_CHECKPOINT(vm)
<!--NeedCopy-->
```

VM_HAS_NO_SUSPEND_VDI

VM cannot be resumed because it has no suspend VDI

Signature:

```
VM_HAS_NO_SUSPEND_VDI(vm)
<!--NeedCopy-->
```

VM_HAS_PCI_ATTACHED

This operation could not be performed, because the VM has one or more PCI devices passed through.

Signature:

```
VM_HAS_PCI_ATTACHED(vm)
<!--NeedCopy-->
```

VM_HAS_SRIOV_VIF

This operation could not be performed, because the VM has one or more SR-IOV VIFs.

Signature:

```
VM_HAS_SRIOV_VIF(vm)
<!--NeedCopy-->
```

VM_HAS_TOO_MANY_SNAPSHOTS

Cannot migrate a VM with more than one snapshot.

Signature:

```
VM_HAS_TOO_MANY_SNAPSHOTS(vm)
<!--NeedCopy-->
```

VM_HAS_VGPU

This operation could not be performed, because the VM has one or more virtual GPUs.

Signature:

```
VM_HAS_VGPU(vm)
<!--NeedCopy-->
```

VM_HAS_VUSBS

The operation is not allowed when the VM has VUSBs.

Signature:

```
VM_HAS_VUSBS(VM)
<!--NeedCopy-->
```

VM_HOST_INCOMPATIBLE_VERSION

This VM operation cannot be performed on an older-versioned host during an upgrade.

Signature:

```
VM_HOST_INCOMPATIBLE_VERSION(host, vm)
<!--NeedCopy-->
```

VM_HOST_INCOMPATIBLE_VERSION_MIGRATE

Cannot migrate a VM to a destination host which is older than the source host.

Signature:

```
VM_HOST_INCOMPATIBLE_VERSION_MIGRATE(host, vm)
<!--NeedCopy-->
```

VM_HOST_INCOMPATIBLE_VIRTUAL_HARDWARE_PLATFORM_VERSION

You attempted to run a VM on a host that cannot provide the VM's required Virtual Hardware Platform version.

Signature:

```
VM_HOST_INCOMPATIBLE_VIRTUAL_HARDWARE_PLATFORM_VERSION(host,
    host_versions, vm, vm_version)
<!--NeedCopy-->
```

VM_HVM_REQUIRED

HVM is required for this operation

Signature:

```
VM_HVM_REQUIRED(vm)
<!--NeedCopy-->
```

VM_INCOMPATIBLE_WITH_THIS_HOST

The VM is incompatible with the CPU features of this host.

Signature:

```
VM_INCOMPATIBLE_WITH_THIS_HOST(vm, host, reason)
<!--NeedCopy-->
```

VM_IS_IMMOBILE

The VM is configured in a way that prevents it from being mobile.

Signature:

```
VM_IS_IMMOBILE(VM)
<!--NeedCopy-->
```

VM_IS_PART_OF_AN_APPLIANCE

This operation is not allowed as the VM is part of an appliance.

Signature:

```
VM_IS_PART_OF_AN_APPLIANCE(vm, appliance)
<!--NeedCopy-->
```

VM_IS_PROTECTED

This operation cannot be performed because the specified VM is protected by HA

Signature:

```
VM_IS_PROTECTED(vm)
<!--NeedCopy-->
```

VM_IS_TEMPLATE

The operation attempted is not valid for a template VM

Signature:

```
VM_IS_TEMPLATE(vm)
<!--NeedCopy-->
```

VM_IS_USING_NESTED_VIRT

This operation is illegal because the VM is using nested virtualization.

Signature:

```
VM_IS_USING_NESTED_VIRT(VM)
<!--NeedCopy-->
```

VM_LACKS_FEATURE

You attempted an operation on a VM which lacks the feature.

Signature:

```
VM_LACKS_FEATURE(vm)
<!--NeedCopy-->
```

VM_LACKS_FEATURE_SHUTDOWN

You attempted an operation which needs the cooperative shutdown feature on a VM which lacks it.

Signature:

```
VM_LACKS_FEATURE_SHUTDOWN(vm)
<!--NeedCopy-->
```

VM_LACKS_FEATURE_STATIC_IP_SETTING

You attempted an operation which needs the VM static-ip-setting feature on a VM which lacks it.

Signature:

```
VM_LACKS_FEATURE_STATIC_IP_SETTING(vm)
<!--NeedCopy-->
```

VM_LACKS_FEATURE_SUSPEND

You attempted an operation which needs the VM cooperative suspend feature on a VM which lacks it.

Signature:

```
VM_LACKS_FEATURE_SUSPEND(vm)
<!--NeedCopy-->
```

VM_LACKS_FEATURE_VCPU_HOTPLUG

You attempted an operation which needs the VM hotplug-vcpu feature on a VM which lacks it.

Signature:

```
VM_LACKS_FEATURE_VCPU_HOTPLUG(vm)
<!--NeedCopy-->
```

VM_MEMORY_SIZE_TOO_LOW

The specified VM has too little memory to be started.

Signature:

```
VM_MEMORY_SIZE_TOO_LOW(vm)
<!--NeedCopy-->
```

VM_MIGRATE_CONTACT_REMOTE_SERVICE_FAILED

Failed to contact service on the destination host.

No parameters.

VM_MIGRATE_FAILED

An error occurred during the migration process.

Signature:

```
VM_MIGRATE_FAILED(vm, source, destination, msg)
<!--NeedCopy-->
```

VM_MISSING_PV_DRIVERS

You attempted an operation on a VM which requires PV drivers to be installed but the drivers were not detected.

Signature:

```
VM_MISSING_PV_DRIVERS(vm)
<!--NeedCopy-->
```

VM_NOT_RESIDENT_HERE

The specified VM is not currently resident on the specified server.

Signature:

```
VM_NOT_RESIDENT_HERE(vm, host)
<!--NeedCopy-->
```

VM_NO_CRASHDUMP_SR

This VM does not have a crash dump SR specified.

Signature:

```
VM_NO_CRASHDUMP_SR(vm)
<!--NeedCopy-->
```

VM_NO_EMPTY_CD_VBD

The VM has no empty CD drive (VBD).

Signature:

```
VM_NO_EMPTY_CD_VBD(vm)
<!--NeedCopy-->
```

VM_NO_SUSPEND_SR

This VM does not have a suspend SR specified.

Signature:

```
VM_NO_SUSPEND_SR(vm)
<!--NeedCopy-->
```

VM_NO_VCPUS

You need at least 1 VCPU to start a VM

Signature:

```
VM_NO_VCPUS(vm)
<!--NeedCopy-->
```

VM_OLD_PV_DRIVERS

You attempted an operation on a VM which requires a more recent version of the PV drivers. Please upgrade your PV drivers.

Signature:

```
VM_OLD_PV_DRIVERS(vm, major, minor)
<!--NeedCopy-->
```

VM_PCI_BUS_FULL

The VM does not have any free PCI slots

Signature:

```
VM_PCI_BUS_FULL(VM)
<!--NeedCopy-->
```

VM_PV_DRIVERS_IN_USE

VM PV drivers still in use

Signature:

```
VM_PV_DRIVERS_IN_USE(vm)
<!--NeedCopy-->
```

VM_REBOOTED

The VM unexpectedly rebooted

Signature:

```
VM_REBOOTED(vm)
<!--NeedCopy-->
```

VM_REQUIRES_GPU

You attempted to run a VM on a host which doesn't have a pGPU available in the GPU group needed by the VM. The VM has a vGPU attached to this GPU group.

Signature:

```
VM_REQUIRES_GPU(vm, GPU_group)
<!--NeedCopy-->
```

VM_REQUIRES_IOMMU

You attempted to run a VM on a host which doesn't have I/O virtualization (IOMMU/VT-d) enabled, which is needed by the VM.

Signature:

```
VM_REQUIRES_IOMMU(host)
<!--NeedCopy-->
```

VM_REQUIRES_NETWORK

You attempted to run a VM on a host which doesn't have a PIF on a Network needed by the VM. The VM has at least one VIF attached to the Network.

Signature:

```
VM_REQUIRES_NETWORK(vm, network)
<!--NeedCopy-->
```

VM_REQUIRES_SR

You attempted to run a VM on a host which doesn't have access to an SR needed by the VM. The VM has at least one VBD attached to a VDI in the SR.

Signature:

```
VM_REQUIRES_SR(vm, sr)
<!--NeedCopy-->
```

VM_REQUIRES_VDI

VM cannot be started because it requires a VDI which cannot be attached

Signature:

```
VM_REQUIRES_VDI(vm, vdi)
<!--NeedCopy-->
```

VM_REQUIRES_VGPU

You attempted to run a VM on a host on which the vGPU required by the VM cannot be allocated on any pGPUs in the GPU_group needed by the VM.

Signature:

```
VM_REQUIRES_VGPU(vm, GPU_group, vGPU_type)
<!--NeedCopy-->
```

VM_REQUIRES_VUSB

You attempted to run a VM on a host on which the VUSB required by the VM cannot be allocated on any PUSBs in the USB_group needed by the VM.

Signature:

```
VM_REQUIRES_VUSB(vm, USB_group)
<!--NeedCopy-->
```

VM_REVERT_FAILED

An error occured while reverting the specified virtual machine to the specified snapshot

Signature:

```
VM_REVERT_FAILED(vm, snapshot)
<!--NeedCopy-->
```

VM_SHUTDOWN_TIMEOUT

VM failed to shutdown before the timeout expired

Signature:

```
VM_SHUTDOWN_TIMEOUT(vm, timeout)
<!--NeedCopy-->
```

VM_SNAPSHOT_WITH_QUIESCE_FAILED

The quiesced-snapshot operation failed for an unexpected reason

Signature:

```
VM_SNAPSHOT_WITH_QUIESCE_FAILED(vm)
<!--NeedCopy-->
```

VM_SNAPSHOT_WITH_QUIESCE_NOT_SUPPORTED

The VSS plug-in is not installed on this virtual machine

Signature:

```
VM_SNAPSHOT_WITH_QUIESCE_NOT_SUPPORTED(vm, error)
<!--NeedCopy-->
```

VM_SNAPSHOT_WITH_QUIESCE_PLUGIN_DEOS_NOT_RESPOND

The VSS plug-in cannot be contacted

Signature:

```
VM_SNAPSHOT_WITH_QUIESCE_PLUGIN_DEOS_NOT_RESPOND(vm)
<!--NeedCopy-->
```

VM_SNAPSHOT_WITH_QUIESCE_TIMEOUT

The VSS plug-in has timed out

Signature:

```
VM_SNAPSHOT_WITH_QUIESCE_TIMEOUT(vm)
<!--NeedCopy-->
```

VM_SUSPEND_TIMEOUT

VM failed to suspend before the timeout expired

Signature:

```
VM_SUSPEND_TIMEOUT(vm, timeout)
<!--NeedCopy-->
```

VM_TOO_MANY_VCPUS

Too many VCPUs to start this VM

Signature:

```
VM_TOO_MANY_VCPUS(vm)
<!--NeedCopy-->
```

VM_TO_IMPORT_IS_NOT_NEWER_VERSION

The VM cannot be imported unforced because it is either the same version or an older version of an existing VM.

Signature:

```
VM_TO_IMPORT_IS_NOT_NEWER_VERSION(vm, existing_version,
    version_to_import)
<!--NeedCopy-->
```

VM_UNSAFE_BOOT

You attempted an operation on a VM that was judged to be unsafe by the server. This can happen if the VM would run on a CPU that has a potentially incompatible set of feature flags to those the VM requires. If you want to override this warning then use the 'force'option.

Signature:

```
VM_UNSAFE_BOOT(vm)
<!--NeedCopy-->
```

VTPM_MAX_AMOUNT_REACHED

The VM cannot be associated with more VTPMs.

Signature:

```
VTPM_MAX_AMOUNT_REACHED(amount)
<!--NeedCopy-->
```

WLB_AUTHENTICATION_FAILED

WLB rejected our configured authentication details.

No parameters.

WLB_CONNECTION_REFUSED

WLB refused a connection to the server.

No parameters.

WLB_CONNECTION_RESET

The connection to the WLB server was reset.

No parameters.

WLB_DISABLED

This pool has wlb-enabled set to false.

No parameters.

WLB_INTERNAL_ERROR

WLB reported an internal error.

No parameters.

WLB_MALFORMED_REQUEST

WLB rejected the server's request as malformed.

No parameters.

WLB_MALFORMED_RESPONSE

WLB said something that the server wasn't expecting or didn't understand. The method called on WLB, a diagnostic reason, and the response from WLB are returned.

Signature:

```
WLB_MALFORMED_RESPONSE(method, reason, response)
<!--NeedCopy-->
```

WLB_NOT_INITIALIZED

No WLB connection is configured.

No parameters.

WLB_TIMEOUT

The communication with the WLB server timed out.

Signature:

```
WLB_TIMEOUT(configured_timeout)
<!--NeedCopy-->
```

WLB_UNKNOWN_HOST

The configured WLB server name failed to resolve in DNS.

No parameters.

WLB_URL_INVALID

The WLB URL is invalid. Ensure it is in the format: <ipaddress>:<port>. The configured/given URL is returned.

Signature:

```
WLB_URL_INVALID(url)
<!--NeedCopy-->
```

WLB_XENSERVER_AUTHENTICATION_FAILED

WLB reported that the server rejected its configured authentication details.

No parameters.

WLB_XENSERVER_CONNECTION_REFUSED

WLB reported that the server refused to let it connect (even though we're connecting perfectly fine in the other direction).

No parameters.

WLB_XENSERVER_MALFORMED_RESPONSE

WLB reported that the server said something to it that WLB wasn't expecting or didn't understand.

No parameters.

WLB_XENSERVER_TIMEOUT

WLB reported that communication with the server timed out.

No parameters.

WLB_XENSERVER_UNKNOWN_HOST

WLB reported that its configured server name for this server instance failed to resolve in DNS.

No parameters.

XAPI_HOOK_FAILED

3rd party xapi hook failed

Signature:

```
XAPI_HOOK_FAILED(hook_name, reason, stdout, exit_code)
<!--NeedCopy-->
```

XENAPI_MISSING_PLUGIN

The requested plug-in could not be found.

Signature:

```
XENAPI_MISSING_PLUGIN(name)
<!--NeedCopy-->
```

XENAPI_PLUGIN_FAILURE

There was a failure communicating with the plug-in.

Signature:

```
XENAPI_PLUGIN_FAILURE(status, stdout, stderr)
<!--NeedCopy-->
```

XEN_INCOMPATIBLE

The current version of Xen or its control libraries is incompatible with the Toolstack.

No parameters.

XEN_VSS_REQ_ERROR_ADDING_VOLUME_TO_SNAPSET_FAILED

Some volumes to be snapshot could not be added to the VSS snapshot set

Signature:

```
XEN_VSS_REQ_ERROR_ADDING_VOLUME_TO_SNAPSET_FAILED(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_CREATING_SNAPSHOT

An attempt to create the snapshots failed

Signature:

```
XEN_VSS_REQ_ERROR_CREATING_SNAPSHOT(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_CREATING_SNAPSHOT_XML_STRING

Could not create the XML string generated by the transportable snapshot

Signature:

```
XEN_VSS_REQ_ERROR_CREATING_SNAPSHOT_XML_STRING(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_INIT_FAILED

Initialization of the VSS requester failed

Signature:

```
XEN_VSS_REQ_ERROR_INIT_FAILED(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_NO_VOLUMES_SUPPORTED

Could not find any volumes supported by the VSS Provider

Signature:

```
XEN_VSS_REQ_ERROR_NO_VOLUMES_SUPPORTED(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_PREPARING_WRITERS

An attempt to prepare VSS writers for the snapshot failed

Signature:

```
XEN_VSS_REQ_ERROR_PREPARING_WRITERS(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_PROV_NOT_LOADED

The VSS Provider is not loaded

Signature:

```
XEN_VSS_REQ_ERROR_PROV_NOT_LOADED(vm, error_code)
<!--NeedCopy-->
```

XEN_VSS_REQ_ERROR_START_SNAPSHOT_SET_FAILED

An attempt to start a new VSS snapshot failed

Signature:

```
XEN_VSS_REQ_ERROR_START_SNAPSHOT_SET_FAILED(vm, error_code)
<!--NeedCopy-->
```

XMLRPC_UNMARSHAL_FAILURE

The server failed to unmarshal the XMLRPC message; it was expecting one element and received something else.

Signature:

```
XMLRPC_UNMARSHAL_FAILURE(expected, received)
<!--NeedCopy-->
```

XenServer Software Development Kit Guide

May 10, 2023

Welcome to the developer's guide for XenServer. Here you will find
the information you need in order to understand and use the Software

Development Kit (SDK) that XenServer provides. This information
will provide you with some of the architectural background and thinking
that underpins the APIs, the tools that have been provided, and how to
quickly get off the ground.

Getting Started

November 21, 2023

XenServer includes a Remote Procedure Call (RPC) based API providing programmatic
access to the extensive set of XenServer management features and
tools. You can call the XenServer Management API from a remote system or
from local to the XenServer host.

It's possible to write applications that use the XenServer Management API directly through raw RPC calls. However, the task of developing third-party applications is greatly simplified by using a language binding. These language bindings expose the individual API calls as first-class functions in the target language. The XenServer SDK provides language bindings for the C, C#, Java, Python, and PowerShell programming languages.

System Requirements and Preparation

The first step towards using the SDK is to install
XenServer. XenServer is available for
download at https://www.xenserver.com/downloads. For detailed instructions on
how to set up your development host, see the Install XenServer.
When the installation is complete, note the *host IP address* and the *host password*.

Downloading

The SDK is packaged as a ZIP file and is available as a free download
from https://www.xenserver.com/downloads.

The Python module is also available as a package on PyPi.
See section SDK Languages - Python for details.

SDK Languages

The extracted contents of the SDK ZIP file are in the `XenServer-SDK` directory. The following is an overview of its structure. Where necessary, subdirectories have their own individual README files.

> **Note:**
>
> The contents of the SDK ZIP consist of library binaries and their source code. Previous releases of the SDK ZIP included a number of examples for each of the SDK languages to help you get started with the SDK. These examples have now been removed from the SDK ZIP and are available at XenServer SDK usage examples on GitHub.
>
> The examples provided aren't the same across all the SDK languages. If you intend to use one language, it's advisable to browse the sample code available in the others as well.

The top level of the `XenServer-SDK` directory includes the XenServer Management API Reference document. This document provides an overview of the API types and classes.

The API supports two wire formats, one based upon XML-RPC and one based upon JSON-RPC (v1.0 and v2.0 are both recognised). For more information on the API semantics and the wire protocol of the RPC messages, see section XenServer Management API. The format supported by each of the SDK languages is specified in the following paragraphs.

C

The `XenServer-SDK` directory contains the following folders that are relevant to C programmers:

- `libxenserver`: The XenServer SDK for C.
  - `bin`: The libxenserver compiled library.
  - `src`: The libxenserver source code and a Makefile to build it. Every API object is associated with a header file, which contains declarations for all the object's API functions. For example, the type definitions and functions required to invoke VM operations are all contained in `xen_vm.h`.

RPC protocol:

The C SDK supports the XML-RPC protocol.

Platform supported:

- Linux
- Windows (under cygwin)

Library:

The library is generated as `libxenserver.so` that is linked by C programs.

> **Note:**
>
> This library is not backwards compatible. To interact with hosts running older versions of XenServer or Citrix Hypervisor, it is advisable to use the library of the same version as the host.

Dependencies:

The library is dependent upon the XML toolkit from the GNOME project, by Daniel Veillard, et al. It is packaged as `libxml2-devel` on CentOS and `libxml2-dev` on Debian.

Building the source code:

To compile the source code type `make` in the `libxenserver/src` directory. To build on Windows with cygwin type `make CYGWIN=1`.

Examples:

Examples on the usage of the C SDK can be found at libxenserver usage examples on GitHub.

C

The `XenServer-SDK` directory contains the following folders that are relevant to C# programmers:

- `XenServer.NET`: The XenServer SDK for C#.NET.
  - `XenServer.NET.x.y.z.nupkg`: The compiled library shipped as a NuGet package.
  - `src`: XenServer.NET source code shipped as a Microsoft Visual Studio project. Every API object is associated with one C# file. For example, the functions implementing the VM operations are contained within the file `VM.cs`.

RPC protocol:

The C# SDK supports the JSON-RPC v2.0 protocol.

Platform supported:

- Windows
- Linux

The compiled library targets .NET Framework version 4.5 and .NET Standard 2.0.

Library:

The library is generated as a Dynamic Link Library `XenServer.dll` that C# programs can reference.

The C# SDK is backwards compatible and can be used to interact with hosts running all versions of XenServer or Citrix Hypervisor (from XenServer 7.3 and Citrix Hypervisor 8.0 onwards).

Dependencies:

Newtonsoft JSON.NET by James Newton-King. We use a patched version of the library (Newtonsoft.Json.CH.dll) shipped with XenServer.NET and recommend that you use this one, although others may work.

Installation:

To use the NuGet package in your code, add its location as a NuGet source:

```
nuget sources Add -Name "Offline Package" -Source "/full/path/to/
    package/directory"
```

Then install the package:

```
nuget install XenServer.NET
```

Building the source code:

Open the project `XenServer.csproj` in Visual Studio 2022. You should now be ready to build the source code.

Examples:

Examples on the usage of the C# SDK can be found at XenServer.NET usage examples on GitHub.

Java

The `XenServer-SDK` directory contains the following folders that are relevant to Java programmers:

- `XenServerJava`: The XenServer SDK for Java
  - `xen-api-x.y.z.jar`: Java archive file containing the compiled library.
  - `xen-api-x.y.z-javadoc.jar`: Java archive file containing the documentation.
  - `xen-api-x.y.z-sources.jar`: Java archive file containing the source code as a Maven project. Every API object is associated with one Java file. For example the functions implementing the VM operations are contained within the file `VM.java`.

RPC protocol:

The Java SDK supports the XML-RPC protocol.

Platform supported:

- Linux
- Windows

Library:

- The library is a Java archive file `xenserver.jar` that is linked by Java programs.

> **Note:**
>
> This library is generally, but not fully, backwards compatible. To interact with hosts running older versions of XenServer or Citrix Hypervisor, it is advisable to use the library of the same version as the host.

Dependencies:

XenServerJava is dependent upon Apache XML-RPC by the Apache Software Foundation.

Examples:

Examples on the usage of the Java SDK can be found at XenServerJava usage examples on GitHub.

PowerShell

The `XenServer-SDK` directory contains the following folders that are relevant to PowerShell users:

- `XenServerPowerShell`: The XenServer SDK for PowerShell.
    - `XenServerPSModule`: The XenServer PowerShell module.
    - `src`: C# source code for the XenServer PowerShell cmdlets.

RPC protocol:

The PowerShell SDK supports the JSON-RPC v2.0 protocol.

Platform supported:

- Windows
- Linux

This library requires .NET 6.0 and PowerShell 7.2 or greater.

Library:

- `XenServerPSModule`

> **Note:**
>
> This module is generally, but not fully, backwards compatible. To interact with hosts running older versions of XenServer or Citrix Hypervisor, it is advisable to use the module of the same version as the host.

Dependencies:

- Newtonsoft JSON.NET by James Newton-King. We use a patched version of the library (Newtonsoft.Json.CH.dll) shipped with the XenServer PowerShell module, although others may work.
- XenServer.NET, the C# SDK for XenServer.

Installation:

1. Extract the contents of the SDK ZIP file.

   > **Note:**
   >
   > Some web browsers may mark the SDK ZIP file as "blocked"during the download. To import the module successfully you will need to unblock the SDK ZIP file before extracting its contents. To unblock the SDK ZIP file, right-click on it and launch the **Properties** dialog. Click the **Unblock** button, then the **Apply** or **OK** button.

2. Navigate to the extracted `XenServer-SDK\XenServerPowerShell` directory and copy the whole folder `XenServerPSModule` into your PowerShell modules directory.

   - On Windows this will normally be `$env:UserProfile\Documents\PowerShell\Modules` for per-user configuration, or `$env:ProgramFiles\PowerShell\7\Modules` for system-wide configuration.
   - On Linux this will be `~/.local/share/powershell/Modules` for per-user configuration, or `/usr/local/share/powershell/Modules` for system-wide configuration.

   For more information see PowerShell's documentation on module paths:

   ```
   Get-Help about_PSModulePath
   <!--NeedCopy-->
   ```

3. Open a PowerShell 7 prompt as administrator.

   To do this, open the Windows **Start** menu by clicking the **Start** icon, find the item **PowerShell 7**, right-click it and select **Run as administrator**.

4. Determine the current execution policy:

   ```
   Get-ExecutionPolicy
   <!--NeedCopy-->
   ```

   If the current policy is `Restricted`, you need to set it to `RemoteSigned`:

   ```
   Set-ExecutionPolicy RemoteSigned
   <!--NeedCopy-->
   ```

   You should understand the security implications of this change. If you are unsure, see PowerShell's documentation on execution policies:

```
Get-Help about_Execution_Policies
<!--NeedCopy-->
```

If the current policy is `AllSigned`, you will be able to use the XenServer PowerShell module, but it will be inconvenient because this policy requires even scripts that you write on the local computer to be signed. You may want to change it to `RemoteSigned`, as above.

If the current policy is `RemoteSigned`, `ByPass`, or `Unrestricted` there is nothing to do.

5. Exit the privileged instance of PowerShell.
6. Open a PowerShell 7 prompt as a regular user (click **Start** > **PowerShell 7**) and import the XenServer PowerShell module:

```
Import-Module XenServerPSModule
<!--NeedCopy-->
```

7. If you wish to load specific environment settings when the XenServer PowerShell module is loaded, create the file `XenServerProfile.ps1` and put it in the folder containing your `$PROFILE` file for per-user configuration, or in `$PSHOME` for system-wide configuration.

   - On Windows these will normally be `$env:UserProfile\Documents\PowerShell` for per-user configuration, or `$env:ProgramFiles\PowerShell\7` for system-wide configuration.
   - On Linux these will be `~/.config/powershell` for per-user configuration, or `/opt/microsoft/powershell/7` for system-wide configuration.

Getting help:

For an overview of the XenServer PowerShell module, type:

```
Get-Help about_XenServer
<!--NeedCopy-->
```

You can obtain a list of all available cmdlets by running:

```
Get-Command -Module XenServerPSModule
<!--NeedCopy-->
```

For help with a specific command use:

```
Get-Help [CommandName]
<!--NeedCopy-->
```

See Using the XenServer PowerShell module for an overview of the module cmdlets and their structure, and a number of walk-throughs on how to perform certain specialized tasks.

Building and debugging the source code:

Open the project `XenServerPowerShell.csproj` in Visual Studio 2022. You should now be ready to build the source code.

If in Debug mode, clicking **Start** will launch a PowerShell 7 prompt as an external process and import the compiled `XenServerPowerShell.dll` as a module (without, however, processing the scripts, types, and formats shipped within the XenServer PowerShell module). You should now be ready to debug the cmdlets.

Examples:

Examples on the usage of the XenServer Powershell module can be found at XenServer PowerShell Module usage examples on GitHub.

Python

The `XenServer-SDK` directory contains the following folders that are relevant to Python developers:

- `XenServerPython`: This directory contains the XenServer Python module *XenAPI.py*.

Alternative installation The Python module is also available as a package on PyPi. To install the package, enable the virtual environment where it will be used and run

```
pip install XenAPI
```

RPC protocol:

The Python module supports the XML-RPC protocol.

Platform supported:

- Linux
- Windows

Library:

- XenAPI.py

Dependencies:

- xmlrpclib

Examples:

Examples on the usage of the XenServer Powershell module can be found at XenAPI.py usage examples on GitHub.

Command line interface (CLI)

Besides using raw RPC or one of the supplied SDK languages, third-party software developers can integrate with XenServer hosts
by using the xe command line interface `xe`. The xe CLI is installed by default on XenServer hosts. A stand-alone remote CLI is also
available for Linux. On Windows, the `xe.exe` CLI executable is
installed along with XenCenter.

Platform supported:

- Linux
- Windows

Library:

- None

Binary:

- xe on Linux
- xe.exe on Windows

Dependencies:

- None

The CLI allows almost every API call to be directly invoked from a
script or other program, silently taking care of the required session
management. The xe CLI syntax and capabilities are described in detail
in the Command line interface documentation.

> **Note:**
>
> When running the CLI from a XenServer host console, tab completion of both command names and arguments is available.

Overview of the XenServer Management API

November 16, 2023

This chapter introduces the XenServer Management API (after here referred to
as the "API") and its associated object model. The API has the following
key features:

- **Management of all aspects of the XenServer host.**

  The API allows you to manage VMs, storage, networking, host configuration, and pools. Performance and status metrics can also be queried from the API.

- **Persistent Object Model.**

  The results of all side-effecting operations (for example: object creation, deletion, and parameter changes) are persisted in a server-side database that is managed by XenServer.

- **An event mechanism.**

  Through the API, clients can register to be notified when persistent (server-side) objects are changed. This enables applications to track datamodel changes performed by concurrently running clients.

- **Synchronous and asynchronous invocation.**

  All API calls can be invoked synchronously (that is, block until completion). Any API call that might be long-running can also be invoked *asynchronously*. Asynchronous calls return immediately with a reference to a *task* object. This task object can be queried (through the API) for progress and status information. When an asynchronously invoked operation completes, the result (or error code) is available from the task object.

- **Remotable and Cross-Platform.**

  The client issuing the API calls doesn't have to be resident on the host being managed. The client also does not have to be connected to the host over ssh to run the API. API calls use the RPC protocol to transmit requests and responses over the network.

- **Secure and Authenticated Access.**

  The RPC API backend running on the host accepts secure socket connections. This allows a client to run the APIs over the https protocol. Further, all the API calls run in the context of a login session generated through user name and password validation at the server. This provides secure and authenticated access to the XenServer installation.

XenServer Management API Deprecation Policy

Items that will be removed in a future release are marked as deprecated.

By default, deprecated APIs and product functionality continue to be supported up to and including the next XenServer Long Term Service Release (LTSR). Deprecated items are usually removed in releases following that LTSR.

In exceptional cases, an item might be deprecated and removed before the next LTSR. For example, a change might be required to improve security. If this happens, customers are made aware of the change to the API or the product functionality.

This deprecation policy applies only to APIs and functionality that are documented at the following locations:

- Product Documentation
- Developer Documentation

Getting Started with the API

Let's start our tour of the API by describing the calls required to create a VM on a XenServer installation, and take it through a start/suspend/resume/stop cycle. This section does not reference code in any specific language. At this stage we just describe the informal sequence of RPC invocations that do our "install and start" task.

> **Note:**
>
> We recommend strongly against using the `VM.create` call, which might be removed or changed in a future version of the API. Read on to learn other ways to make a new VM.

Authentication: acquiring a session reference

The first step is to call `Session.login_with_password(username, password, client_API_version, originator)`. The API is session based, so before you can make other calls you must authenticate with the server. Assuming the user name and password are authenticated correctly, the result of this call is a *session reference*. Subsequent API calls take the session reference as a parameter. In this way, we ensure that only API users who are suitably authorized can perform operations on a XenServer installation. You can continue to use the same session for any number of

API calls. When you have finished the session, it is recommended that you call `Session.logout(session)` to clean up (see section `Logging out`).

Acquiring a list of templates to base a new VM installation on

The next step is to query the list of "templates"on the host. Templates are specially marked VM objects that specify suitable default parameters for various supported guest types. (If you want to see a quick enumeration of the templates on a XenServer installation for yourself, you can run the `xe template-list` CLI command.) To get a list of templates from the API, find the VM objects on the server that have their `is_a_template` field set to true. One way to do find these objects is by calling `VM.get_all_records(session)` where the session parameter is the reference we acquired from our `Session.login_with_password` call earlier. This call queries the server, returning a snapshot (taken at the time of the call) containing all the VM object references and their field values.

(Remember that at this stage we are not concerned with how the returned object references and field values can be manipulated in any particular client language: that detail is dealt with by each language-specific SDK and described concretely in the following chapter. For now, assume the existence of an abstract mechanism for reading and manipulating objects and field values returned by API calls.)

Now we have a snapshot of the VM objects'field values in the memory of our client application, iterate through them and find the VMs that have their `is_a_template` value set to **`true`**. At this stage, let's assume that our example application further iterates through the template objects and remembers the reference corresponding to the one that has its "`name_label`"set to "Debian Buster 10"(one of the default Linux templates supplied with XenServer).

Installing the VM based on a template

Continuing through our example, we must now install a new VM based on the template we selected. The installation process requires two API calls:

- First we must now invoke the API call `VM.clone(session, t_ref, "my first VM")`. This call tells the server to clone the

VM object
referenced by `t_ref` to make a new VM object. The return
value of this call is the VM reference corresponding to the
newly created VM. Let's call this `new_vm_ref`.

- At this stage, the object referred to by `new_vm_ref` is still a
template (like the VM object referred to by `t_ref`, from which
it was cloned). To make `new_vm_ref` into a VM object, we must
call `VM.provision(session, new_vm_ref)`. When this call returns the
`new_vm_ref` object will have had its `is_a_template` field set to
false, indicating that `new_vm_ref` now refers to a regular VM ready
for starting.

> **Note:**
>
> The provision operation can take a few minutes, as it is during this call that the template's disk images are created.
> For the Debian template, the newly created disks are also at this stage populated with a Debian root filesystem.

Taking the VM through a start/suspend/resume/stop cycle

Now we have an object reference representing our newly installed VM, it
is trivial to take it through a few lifecycle operations:

- To start our VM, we can call `VM.start(session, new_vm_ref)`
- After it's running, we can suspend it by calling
`VM.suspend(session, new_vm_ref)`,
- We can resume it by calling `VM.resume(session, new_vm_ref)`.
- We can call `VM.shutdown(session, new_vm_ref)` to shut down the VM
cleanly.

Logging out

When an application is finished interacting with a XenServer host,
it is good practice to call `Session.logout(session)`. This call invalidates
the session reference (so it cannot be used in subsequent API calls) and
deallocates server-side memory used to store the session object.

Although inactive sessions eventually time out, the server has a
hardcoded limit of 500 concurrent sessions for each `username` or

`originator`. After this limit has been reached, fresh logins evict the session objects that have been used least recently. The session references of these evicted session objects become invalid. For successful interoperability with other applications, concurrently accessing the server, the best policy is:

- Choose a string that identifies your application and its version.
- Create a single session at start-of-day, using that identifying string for the `originator` parameter to `Session.login_with_password`.
- Use this session throughout the application and then explicitly logout when possible.

> **Note:**
>
> Sessions can be used across multiple separate client-server *network connections*.

If a poorly written client leaks sessions or otherwise exceeds the limit, then as long as the client uses an appropriate `originator` argument, it is easily identifiable from the XenServer logs. XenServer destroys the longest-idle sessions of the rogue client only. This behavior might cause problems for that client but not for other clients. If the misbehaving client doesn't specify an `originator`, it is harder to identify and causes the premature destruction of other client sessions that also didn't specify an `originator`.

Install and start example: summary

We have seen how the API can be used to install a VM from a XenServer template and perform various lifecycle operations on it. Note that the number of calls we had to make to affect these operations was small:

- One call to acquire a session: `Session.login_with_password()`
- One call to query the VM (and template) objects present on the XenServer installation: `VM.get_all_records()`. Recall that we used the information returned from this call to select a suitable template to install from.
- Two calls to install a VM from our chosen template: `VM.clone()`, followed by `VM.provision()`.
- One call to start the resultant VM: `VM.start()` (and similarly other single calls to suspend, resume, and shut down accordingly)

- And then one call to log out `Session.logout()`.

Although the API as a whole is complex and fully featured, common tasks (such as VM lifecycle operations) are straightforward, requiring only a few simple API calls.
Keep this fact in mind as you study the next section which might, on first reading, appear a little daunting!

Object Model Overview

This section gives a high-level overview of the object model of the API.
For a more detailed description of the parameters and methods of each class, see the XenServer Management API reference.

We start by giving a brief outline of some of the core classes that make up the API.
(Don't worry if these definitions seem abstract in their initial presentation. The textual description in the following sections, and the code-sample walk through in the next section make these concepts concrete.)

VM

A VM object represents a particular Virtual Machine instance on a XenServer host or resource pool. Example methods include `start`, `suspend`, `pool_migrate`. Example parameters include `power_state`, `memory_static_max`, and `name_label`. (In the previous section we saw how the VM class is used to represent both templates and regular VMs).

Host

A host object represents a physical host in a XenServer pool. Example methods include `reboot` and `shutdown`. Example parameters include `software_version`, `hostname`, and [IP]`address`.

VDI

A VDI object represents a Virtual Disk Image. Virtual Disk Images can be attached to VMs. A
block device appears inside the VM through which the bits encapsulated by the attached Virtual Disk Image can be read
and written. Example methods of the VDI class include `resize` and `clone`. Example fields include `virtual_size` and `sharable`.
When we called `VM.provision` on the VM template in our previous example, some VDI objects were

automatically created to
represent the newly created disks. These VDIs were attached to the VM object.

SR

An SR object (Storage Repository) aggregates a collection of VDIs, It encapsulates the properties of physical
storage on which the VDIs'bits reside. Example parameters include:

- `type` which determines the storage-specific driver a XenServer installation uses to read/write the SR's VDIs
- `physical_utilisation`

Example methods include

- `scan` which invokes the storage-specific driver to acquire a list of the VDIs contained with the SR and the properties of these VDIs
- `create` which initializes a block of physical storage so it is ready to store VDIs

Network

A network object
represents a layer-2 network that exists in the environment in which the XenServer host
instance lives. Since XenServer does not manage networks directly, network is a lightweight class that models
physical and virtual network topology. VM and Host objects that are *attached* to a particular
Network object can send network packets to each other. The objects are attached through VIF and PIF instances. For more information, see the following section.

If you are finding this enumeration of classes rather terse, you can skip to the code walk-throughs of the next
chapter. There are plenty of useful applications that can be written
using only a subset of the classes already described. If you want
to continue this description of classes in the abstract, read on.

In addition to the classes listed in the previous section, there are four more that act as
connectors. These *connectors* specify relationships between VMs and Hosts and
Storage and Networks.
The first two of these classes that we consider, *VBD* and *VIF*, determine how VMs are attached to virtual disks and network objects respectively.

VBD

A VBD object (Virtual Block Device) represents an
attachment between a VM and a VDI. When a VM is booted,
its VBD objects are queried to determine which disk
images (VDIs) to attach.

Example methods of the VBD class include:

- `plug` which *hot plugs* a disk device into a running VM, making the specified VDI accessible therein
- `unplug` which *hot unplugs* a disk device from a running guest

Example fields include:

- `device` which determines the device name inside the guest under which the specified VDI is made accessible

VIF

A VIF object (Virtual Network Interface) represents
an attachment between a VM and a Network object. When a VM is booted, its VIF objects are queried to
determine which network devices to create.

Example methods of the VIF class include:

- `plug` which *hot plugs* a network device into a running VM
- `unplug` which *hot unplugs* a network device from a running guest

---

The second set of "connector classes"that we consider determine
how Hosts are attached to Networks and Storage.

PIF

A PIF object (Physical Interface) represents an attachment
between a Host and a Network object. If a host is connected to a
Network over a PIF, packets from the specified host can be
transmitted/received by the corresponding host.
Example fields of the PIF class include:

- `device` which specifies the device name to which the PIF corresponds. For example, *eth0*
- `MAC` which specifies the MAC address of the underlying NIC that a PIF represents

PIFs abstract both physical interfaces and VLANs (the latter distinguished by the existence of a positive integer in the "VLAN"field).

PBD

A PBD object (Physical Block Device) represents an attachment
between a Host and an SR object. Fields include:

- `currently-attached` which specifies whether the chunk of storage represented by the specified SR object is available to the host
- `device_config` which specifies storage-driver specific parameters that determine how the low-level storage devices are configured on the specified host.
  For example, when an SR rendered on an NFS filer, device_config can specify the host-name of the filer and the path on the filer in which the SR files live.

This figure presents a graphical overview of the API classes involved in managing VMs, Hosts, Storage, and Networking.
From this diagram, the symmetry between storage and network configuration, and also the symmetry between virtual machine and host
configuration is plain to see.

Working with VIFs and VBDs

In this section we walk through a few more complex scenarios. These scenarios describe
how various tasks involving virtual storage and network devices can be done using the API.

Creating disks and attaching them to VMs

Let's start by considering how to make a new blank disk image and attach
it to a running VM. We assume that we already have a
running VM, and we know its corresponding API object reference. For example, we
might have created this VM using the procedure described in the previous
section and had the server return its reference to us.
We also assume that we have authenticated with the XenServer installation
and have a corresponding `session reference`. Indeed, the rest of this chapter, for the sake of
brevity, does not mention sessions altogether.

Creating a new blank disk image First, instantiate the disk image on physical storage by calling
`VDI.create()`.
The `VDI.create` call takes a number of parameters, including:

- `name_label` and `name_description`: a human-readable
  name/description for the disk (for example, for convenient display in the UI). These fields can
  be left blank if desired.
- `SR`: the object reference of the Storage Repository representing
  the physical storage in which the VDI's bits are placed.
- `read_only`: setting this field to true indicates that the VDI can
  only be attached to VMs in a read-only fashion. (Attempting to
  attach a VDI with its `read_only` field set to true in a read/write fashion results
  in error.)

Invoking the `VDI.create` call causes the XenServer installation to
create a blank disk image on physical storage, create an associated VDI
object (the datamodel instance that refers to the disk image on physical
storage) and return a reference to this newly created VDI object.

The way in which the disk image is represented on physical storage
depends on the type of the SR in which the created VDI resides. For
example, if the SR is of type "lvm"then the new disk image will be
rendered as an LVM volume; if the SR is of type "nfs"then the new disk
image will be a sparse VHD file created on an NFS filer. (You can query
the SR type through the API using the `SR.get_type()` call.)

> **Note:**
>
> Some SR types might round up the `virtual-size` value to make it divisible by a configured block size.

Attaching the disk image to a VM So far we have a running VM (that we assumed the existence of at the
start of this example) and a fresh VDI that we just created. Right now,
these are both independent objects that exist on the XenServer
Host, but there is nothing linking them together. So our next step is to
create such a link, associating the VDI with our VM.

The attachment is formed by creating a new "connector"object called a
VBD (Virtual Block Device). To create our VBD we invoke the
`VBD.create()` call. The `VBD.create()` call takes a number of parameters including:

- `VM` - the object reference of the VM to which the VDI is to be
  attached
- `VDI` - the object reference of the VDI that is to be attached
- `mode` - specifies whether the VDI is to be attached in a read-only
  or a read-write fashion
- `userdevice` - specifies the block device inside the guest through
  which applications running inside the VM will be able to read/write
  the VDI's bits.
- `type` - specifies whether the VDI is presented inside the VM
  as a regular disk or as a CD. (Note that this particular field has
  more meaning for Windows VMs than it does for Linux VMs, but we will
  not explore this level of detail in this chapter.)

Invoking `VBD.create` makes a VBD object on the XenServer
installation and returns its object reference. However, this call in
itself does not have any side-effects on the running VM (that is, if you
go and look inside the running VM you will see that the block device has
not been created). The fact that the VBD object exists but that the
block device in the guest is not active, is reflected by the fact that
the VBD object's `currently_attached` field is set to false.

For expository purposes, this figure presents
a graphical example that shows the relationship between VMs, VBDs, VDIs
and SRs. In this instance a VM object has 2 attached VDIs: there are two
VBD objects that form the connections between the VM object and its
VDIs; and the VDIs reside within the same SR.

Hotplugging the VBD If we rebooted the VM at this stage then, after rebooting, the block
device corresponding to the VBD would appear: on boot, XenServer

queries all VBDs of a VM and actively attaches each of the corresponding VDIs.

Rebooting the VM is all very well, but recall that we wanted to attach a newly created blank disk to a *running* VM. This can be achieved by invoking the `plug` method on the newly created VBD object. When the `plug` call returns successfully, the block device to which the VBD relates will have appeared inside the running VM –i.e. from the perspective of the running VM, the guest operating system is led to believe that a new disk device has just been *hot plugged*. Mirroring this fact in the managed world of the API, the `currently_attached` field of the VBD is set to true.

Unsurprisingly, the VBD `plug` method has a dual called "`unplug`". Invoking the `unplug` method on a VBD object causes the associated block device to be *hot unplugged* from a running VM, setting the `currently_attached` field of the VBD object to false accordingly.

Creating and attaching Network Devices to VMs

The API calls involved in configuring virtual network interfaces in VMs are similar in many respects to the calls involved in configuring virtual disk devices. For this reason we will not run through a full example of how one can create network interfaces using the API object-model; instead we will use this section just to outline briefly the symmetry between virtual *networking device* and virtual *storage device* configuration.

The networking analogue of the VBD class is the VIF class. Just as a VBD is the API representation of a block device inside a VM, a VIF (*Virtual network InterFace*) is the API representation of a network device inside a VM. Whereas VBDs associate VM objects with VDI objects, VIFs associate VM objects with Network objects. Just like VBDs, VIFs have a `currently_attached` field that determines whether or not the network device (inside the guest) associated with the VIF is currently active or not. And as we saw with VBDs, at VM boot-time the VIFs of the VM are queried and a corresponding network device for each created inside the booting VM. Similarly, VIFs also have `unplug` and `unplug` methods for hot plugging/unplugging network devices in/out of
running VMs.

Host configuration for networking and storage

We have seen that the VBD and VIF classes are used to manage configuration of block devices and network devices (respectively) inside VMs. To manage host configuration of storage and networking there are two analogous classes: PBD (Physical Block Device) and PIF (Physical [network] Interface).

Host storage configuration: PBDs Let us start by considering the PBD class. A `PBD.create()` call takes a
number of parameters including:

Parameter	Description
host	physical machine on which the PBD is available
SR	the Storage Repository that the PBD connects to
device_config	a string-to-string map that is provided to the host's SR-backend-driver, containing the low-level parameters required to configure the physical storage device(s) on which the SR is to be realized. The specific contents of the `device_config` field depend on the type of the SR to which the PBD is connected. (Running `xe sm-list` will show a list of possible SR types; the *configuration* field in this enumeration specifies the `device_config` parameters that each SR type expects.)

For example, imagine we have an SR object *s* of type "nfs" (representing a directory on an NFS filer within which VDIs are stored as VHD files); and let's say that we want a host, *h*, to be able to access *s*. In this case we invoke `PBD.create()` specifying host *h*, SR *s*, and a value for the *device_config* parameter that is the following map:

```
("server", "my_nfs_server.example.com"), ("serverpath", "/scratch/
mysrs/sr1")
```

This tells the XenServer host that SR *s* is accessible on host *h*, and further that to access SR *s*, the host needs to mount the directory `/scratch/mysrs/sr1` on the NFS server named `my_nfs_server.example.com`.

Like VBD objects, PBD objects also have a field called `currently_attached`. Storage repositories can be attached
and detached from a given host by invoking `PBD.plug` and `PBD.unplug`
methods respectively.

Host networking configuration: PIFs Host network configuration is specified by virtue of PIF objects. If a
PIF object connects a network object, *n*, to a host object *h*, then
the network corresponding to *n* is bridged onto a physical interface
(or a physical interface plus a VLAN tag) specified by the fields of the
PIF object.

For example, imagine a PIF object exists connecting host *h* to a
network *n*, and that `device` field of the PIF object is set to `eth0`.
This means that all packets on network *n* are bridged to the NIC in the
host corresponding to host network device `eth0`.

Importing and Exporting VMs

VMs can be exported to a file and later imported to any XenServer host.
Because the import and export operations can take some time to complete,
they are performed using an asynchronous HTTP(S) interface exposed by
XenServer.

XenServer supports a VM input format called XVA.
This is a format specific to Xen-based hypervisors and packages a
single VM as a single file archive consisting of a metadata descriptor
file and disk images.

Detailed information on how to use the HTTP(S) interface to import and
export VMs, and a description of the XVA file format can be found in
chapter Using HTTP to interact with XenServer.

> **Note:**
>
> Single VMs and virtual appliances (vApps) of multiple VMs can be
> exported to and imported from OVF/OVA packages using XenCenter. For
> more details visit chapter
> Importing and Exporting VMs
> of XenCenter's documentation.

Where to look next

In this chapter we have presented a brief high-level overview of the API and its object-model. The aim here is not to present the detailed semantics of the API, but just to provide enough background for you to start reading the code samples of the next chapter and to find your way around the more detailed XenServer Management API reference.

There are a number of places you can find more information:

- The Command line interface documentation contains an overview of the `xe` CLI. Since a good deal of `xe` commands are a thin veneer over the API, playing with `xe` is a good way to start finding your way around the API object model described in this chapter.
- The code samples in the next chapter provide some concrete instances of API coding in a variety of client languages.
- The XenServer Management API reference provides a more detailed description of the API semantics as well as the wire protocol of the RPC messages.
- There are a few scripts that use the API in the XenServer host dom0 itself under `/opt/xensource/libexec/`.
- There is a number of examples for each of the SDK languages at XenServer SDK usage examples on GitHub. These include ready-to-run programs demonstrating operations like creating a VM and taking it through a start/suspend/resume/stop cycle, monitoring events, creating a shared SR, migrating a VM between servers in a pool, etc.

Using the API

November 16, 2023

This chapter describes how to use the XenServer Management API to write real programs to manage XenServer hosts and VMs. The chapter begins with a walk-through of a typical client application and demonstrates how the API can be used to perform common tasks. Example code fragments are given in Python syntax but equivalent code in the other programming

languages would look very similar. The chapter finishes with
walk-throughs of two complete examples.

Anatomy of a typical application

This section describes the structure of a typical application using the
XenServer Management API. Most client applications begin by
connecting to a XenServer host and authenticating using a
username and password. Assuming the authentication succeeds, the server
will create a "session"object and return a reference to the client.
This reference will be passed as an argument to all future API calls.
Once authenticated, the client may search for references to other useful
objects (XenServer hosts, VMs, SRs, and so on) and invoke operations on
them. Operations may be invoked either synchronously or asynchronously;
special task objects represent the state and progress of asynchronous
operations. These application elements are all described in detail in
the following sections.

Choosing a low-level transport

API calls can be issued over two transports:

- SSL-encrypted TCP on port 443 (https) over an IP network
- plaintext over a local Unix domain socket: `/var/xapi/xapi`

Switching from the XML-RPC to the JSON-RPC backend can be done by adding the suffix `/jsonrpc` to the host URL path.

The SSL-encrypted TCP transport is used for all off-host traffic, while the Unix domain socket can be used from services running directly on the XenServer host itself. In the SSL-encrypted TCP transport, all API calls must be directed at the pool coordinator. If directed at a pool supporter, the calls will fail with the error `HOST_IS_SLAVE`, which includes the IP address of the coordinator as an error parameter.

Because the coordinator host of a pool can change, especially if HA is enabled on a pool, clients must implement the following steps to detect a coordinator host change and connect to the new coordinator as required:

Handling pool coordinator changes

1. Subscribe to updates in the list of hosts servers, and maintain a current list of hosts in the pool

2. If the connection to the pool coordinator fails to respond, attempt to connect to all hosts in the list until one responds
3. The first host to respond will return the `HOST_IS_SLAVE` error message, which contains the identity of the new pool coordinator (unless of course the host is the new coordinator)
4. Connect to the new coordinator

> **Note:**
>
> As a special-case, all messages sent through the Unix domain socket are transparently forwarded to the correct node.

Authentication and session handling

The vast majority of API calls take a session reference as their first parameter; failure to supply a valid reference will result in a `SESSION_INVALID` error being returned. Acquire a session reference by supplying a user name and password to the `login_with_password` function.

> **Note:**
>
> As a special-case, if this call is run over the local Unix domain socket then the user name and password are ignored and the call always succeeds.

Every session has an associated "last active"timestamp which is updated on every API call. The server software currently has a built-in limit of 500 active sessions and will remove those with the oldest "last active" field if this limit is exceeded for a given `username` or `originator`. In addition all sessions whose "last active"field is older than 24 hours are also removed. Therefore it is important to:

- Specify an appropriate `originator` when logging in; and
- Remember to log out of active sessions to avoid leaking them; and
- Be prepared to log in again to the server if a `SESSION_INVALID` error is caught.

> **Note:**
>
> A session reference obtained by a login request to the XML-RPC backend can be used in subsequent requests to the JSON-RPC backend, and vice-versa.

In the following Python fragment a connection is established over the Unix domain socket and a session is created:

```
import XenAPI

session = XenAPI.xapi_local()
try:
    session.xenapi.login_with_password("root", "", "2.20", "My Widget
        v0.1")
    ...
finally:
    session.xenapi.session.logout()
<!--NeedCopy-->
```

Finding references to useful objects

Once an application has authenticated the next step is to acquire references to objects in order to query their state or invoke operations on them. All objects have a set of "implicit"messages which include the following:

- `get_by_name_label` : return a list of all objects of a particular class with a particular label;
- `get_by_uuid` : return a single object named by its UUID;
- `get_all` : return a set of references to all objects of a particular class; and
- `get_all_records` : return a map of reference to records for each object of a particular class.

For example, to list all hosts:

```
hosts = session.xenapi.host.get_all()
<!--NeedCopy-->
```

To find all VMs with the name "my first VM":

```
vms = session.xenapi.VM.get_by_name_label('my first VM')
<!--NeedCopy-->
```

> **Note:**
>
> Object `name_label` fields are not guaranteed to be unique and so the `get_by_name_label` API call returns a set of references rather than a single reference.

In addition to the methods of finding objects described above, most objects also contain references to other objects within fields. For

example it is possible to find the set of VMs running on a particular
host by calling:

```
vms = session.xenapi.host.get_resident_VMs(host)
<!--NeedCopy-->
```

Invoking synchronous operations on objects

Once object references have been acquired, operations may be invoked on
them. For example to start a VM:

```
session.xenapi.VM.start(vm, False, False)
<!--NeedCopy-->
```

All API calls are by default synchronous and will not return until the
operation has completed or failed. For example in the case of `VM.start`
the call does not return until the VM has started booting.

> **Note:**
>
> When the `VM.start` call returns, the VM will be booting. To determine
> when the booting has finished, wait for the in-guest agent to report
> internal metrics through the `VM_guest_metrics` object.

Using Tasks to manage asynchronous operations

To simplify managing operations which take quite a long time (for example,
`VM.clone` and `VM.copy`) functions are available in two forms:
synchronous (the default) and asynchronous. Each asynchronous function
returns a reference to a task object which contains information about
the in-progress operation including:

- whether it is pending
- whether it is has succeeded or failed
- progress (in the range 0-1)
- the result or error code returned by the operation

An application which wanted to track the progress of a `VM.clone`
operation and display a progress bar would have code like the following:

```
vm = session.xenapi.VM.get_by_name_label('my vm')
task = session.xenapi.Async.VM.clone(vm)
while session.xenapi.task.get_status(task) == "pending":
    progress = session.xenapi.task.get_progress(task)
    update_progress_bar(progress)
    time.sleep(1)
session.xenapi.task.destroy(task)
<!--NeedCopy-->
```

> **Note:**
>
> A well-behaved client must delete tasks created by asynchronous operations when it has finished reading the result or error.
> If the number of tasks exceeds a built-in threshold then the server will delete the oldest of the completed tasks.

Subscribing to and listening for events

With the exception of the task and metrics classes, whenever an object is modified the server generates an event. Clients can subscribe to this event stream on a per-class basis and receive updates rather than resorting to frequent polling. Events come in three types:

- `add` - generated when an object has been created;
- `del` - generated immediately before an object is destroyed; and
- `mod` - generated when an object's field has changed.

Events also contain a monotonically increasing ID, the name of the class of object, and a snapshot of the object state equivalent to the result of a `get_record()`.

Clients can receive events by calling
`event.from()`
with a list of class names or the special string `*` (to receive events for all classes). The output of `event.from()` includes a token, which can be passed into a subsequent `event.from()` call to receive only the events that have occurred since the last call. If an empty string is passed, `event.from()` will return all events.

The following Python code fragment demonstrates how to print a summary of events for all classes generated by a system:

```
token = ''
fmt = "%8s %s %20s  %5s  %s %s"

while True:
    try
        output = session.xenapi.event_from("*", token, 30.0)
```

```
            events = output['events']
            token = output['token']

            for event in events:
                name = "n/a"
                snapshot = ''
                if "snapshot" in event.keys():
                    snapshot = event['snapshot']
                    if "name_label" in snapshot.keys():
                        name = snapshot['name_label']
                print fmt % (event['id'], event["ref"], event['class'],
                   event['operation'], name, repr(snapshot))

            time.sleep(10)
    finally:
        session.xenapi.session.logout()
<!--NeedCopy-->
```

The full script can be found at `watch-all-events.py` on Github.

Complete application examples

This section describes two complete examples of real programs using the API.

Simultaneously migrating VMs using live migration

This python example (the full script can be found at
`permute.py` on Github)
demonstrates how to use live migration to move VMs simultaneously between
hosts in a Resource Pool. The example makes use of asynchronous API
calls and shows how to wait for a set of tasks to complete.

The program begins with some standard boilerplate and imports the API module

```
import sys, time
import XenAPI
<!--NeedCopy-->
```

Next the commandline arguments containing a server URL, user name,
password and a number of iterations are parsed. The user name and
password are used to establish a session which is passed to the function
`main`, which is called multiple times in a loop. Note the use of
`try: finally:` to make sure the program logs out of its session at the
end.

```
if __name__ == "__main__":
    if len(sys.argv) <> 5:
        print "Usage:"
        print sys.argv[0], " <url> <username> <password> <iterations>"
        sys.exit(1)
    url = sys.argv[1]
    username = sys.argv[2]
    password = sys.argv[3]
    iterations = int(sys.argv[4])
    # First acquire a valid session by logging in:
    session = XenAPI.Session(url)
    session.xenapi.login_with_password(username, password, "2.3",
                                       "Example migration-demo v0.1")
    try:
        for i in range(iterations):
            main(session, i)
    finally:
        session.xenapi.session.logout()
<!--NeedCopy-->
```

The `main` function examines each running VM in the system, taking care to filter out *control domains* (which are part of the system and not controllable by the user). A list of running VMs and their current hosts is constructed.

```
def main(session, iteration):
    # Find a non-template VM object
    all = session.xenapi.VM.get_all()
    vms = []
    hosts = []
    for vm in all:
        record = session.xenapi.VM.get_record(vm)
        if not(record["is_a_template"]) and \
            not(record["is_control_domain"]) and \
            record["power_state"] == "Running":
            vms.append(vm)
            hosts.append(record["resident_on"])
    print "%d: Found %d suitable running VMs" % (iteration, len(vms))
<!--NeedCopy-->
```

Next the list of hosts is rotated:

```
# use a rotation as a permutation
hosts = [hosts[-1]] + hosts[:(len(hosts)-1)]
<!--NeedCopy-->
```

Each VM is then moved using live migration to the new host under this rotation (that is, a VM running on host at position 2 in the list is moved to the host at position 1 in the list, and so on.) In order to execute each of the movements in parallel, the asynchronous version of the `VM.pool_migrate` is used and a list of task references constructed. Note the `live` flag passed to the `VM.pool_migrate`; this causes the VMs to be moved while they are still running.

```
tasks = []
    for i in range(0, len(vms)):
        vm = vms[i]
        host = hosts[i]
        task = session.xenapi.Async.VM.pool_migrate(vm, host, {
  "live": "true"  }
 )
        tasks.append(task)
<!--NeedCopy-->
```

The list of tasks is then polled for completion:

```
finished = False
    records = {
  }

    while not(finished):
        finished = True
        for task in tasks:
            record = session.xenapi.task.get_record(task)
            records[task] = record
            if record["status"] == "pending":
                finished = False
        time.sleep(1)
<!--NeedCopy-->
```

Once all tasks have left the *pending* state (i.e. they have successfully completed, failed or been cancelled) the tasks are polled once more to see if they all succeeded:

```
allok = True
    for task in tasks:
        record = records[task]
        if record["status"] <> "success":
            allok = False
<!--NeedCopy-->
```

If any one of the tasks failed then details are printed, an exception is raised and the task objects left around for further inspection. If all tasks succeeded then the task objects are destroyed and the function returns.

```
if not(allok):
        print "One of the tasks didn't succeed at", \
            time.strftime("%F:%HT%M:%SZ", time.gmtime())
        idx = 0
        for task in tasks:
            record = records[task]
            vm_name = session.xenapi.VM.get_name_label(vms[idx])
            host_name = session.xenapi.host.get_name_label(hosts[idx])
            print "%s : %12s %s -> %s [ status: %s; result = %s; error
                = %s ]" % \
                    (record["uuid"], record["name_label"], vm_name,
                        host_name,       \
                    record["status"], record["result"], repr(record["
                        error_info"]))
            idx = idx + 1
        raise "Task failed"
    else:
        for task in tasks:
            session.xenapi.task.destroy(task)
<!--NeedCopy-->
```

Cloning a VM using the xe CLI

This example is a bash script which uses the xe CLI to clone a VM taking care to shut it down first if it is powered on.

The example begins with some boilerplate which first checks if the environment variable XE has been set: if it has it assumes that it points to the full path of the CLI, else it is assumed that the xe CLI is on the current path. Next the script prompts the user for a server name, user name and password:

```
# Allow the path to the 'xe' binary to be overridden by the XE
    environment variable
if [ -z "${
 XE }
 " ]; then
    XE=xe
fi

if [ ! -e "${
 HOME }
 /.xe" ]; then
    read -p "Server name: " SERVER
    read -p "Username: " USERNAME
    read -p "Password: " PASSWORD
    XE="${
 XE }
  -s ${
 SERVER }
  -u ${
 USERNAME }
  -pw ${
 PASSWORD }
 "
fi
<!--NeedCopy-->
```

Next the script checks its commandline arguments. It requires exactly one: the UUID of the VM which is to be cloned:

```
# Check if there's a VM by the uuid specified
${
 XE }
  vm-list params=uuid | grep -q " ${
 vmuuid }
 $"
if [ $? -ne 0 ]; then
        echo "error: no vm uuid \"${
 vmuuid }
 \" found"
        exit 2
fi
<!--NeedCopy-->
```

The script then checks the power state of the VM and if it is running, it attempts a clean shutdown. The event system is used to wait for the VM to enter state "Halted".

> **Note:**
>
> The xe CLI supports a command-line argument `--minimal` which causes it to print its output without excess whitespace or formatting, ideal for use from scripts. If multiple values are returned they are comma-separated.

```
# Check the power state of the vm
name=$(${
 XE }
  vm-list uuid=${
 vmuuid }
  params=name-label --minimal)
state=$(${
 XE }
  vm-list uuid=${
 vmuuid }
  params=power-state --minimal)
wasrunning=0

# If the VM state is running, we shutdown the vm first
if [ "${
 state }
 " = "running" ]; then
        ${
 XE }
  vm-shutdown uuid=${
 vmuuid }

        ${
```

```
 XE }
  event-wait class=vm power-state=halted uuid=${
 vmuuid }

       wasrunning=1
fi
<!--NeedCopy-->
```

The VM is then cloned and the new VM has its `name_label` set to `cloned_vm`.

```
# Clone the VM
newuuid=$(${
 XE }
  vm-clone uuid=${
 vmuuid }
  new-name-label=cloned_vm)
<!--NeedCopy-->
```

Finally, if the original VM had been running and was shutdown, both it and the new VM are started.

```
# If the VM state was running before cloning, we start it again
# along with the new VM.
if [ "$wasrunning" -eq 1 ]; then
        ${
 XE }
  vm-start uuid=${
 vmuuid }

        ${
 XE }
  vm-start uuid=${
 newuuid }

fi
<!--NeedCopy-->
```

Using HTTP to interact with XenServer

December 9, 2023

XenServer exposes an HTTP(S) interface on each host that can be used to perform various operations.

Each operation is performed by constructing an HTTP(S) GET or POST call to a dedicated URL path. Parameters are appended to the URL following a question mark (?) and separated by ampersands (&). The calls require authentication by providing a valid management API session reference as a query parameter. For example:

```
wget https://<server>/services?session_id=<session_opaque_ref>
<!--NeedCopy-->
```

> **Note:**
>
> Basic auth using a username and password is also available:
>
> ```
> wget https://username:password@<server>/services
> <!--NeedCopy-->
> ```
>
> However, this method is not recommended.

The following sections describe in greater detail some of the available operations.

Importing and exporting VMs as XVA packages

Because the import and export of VMs can take some time to complete, an asynchronous HTTP(S) interface is provided for these operations.

VM Export

To export a VM as an XVA package using the XenServer Management API, construct an HTTP(S) GET call. The call accepts the following query parameters:

Parameter	Description
`session_id`	The opaque reference of the session being used to authenticate.
`uuid`	The `uuid` of the VM; required only if not using the `ref` parameter.
`ref`	The opaque reference of the VM; required only if not using the `uuid` parameter.
`task_id`	(Optional) The opaque reference of a task object with which to monitor the progress of the operation; the task object needs to be created first.

For example, to export a VM to the XVA package `vm.xva` using the Linux command line utility cURL:

```
curl https://server/export?session_id=<session_opaque_ref>&task_id=<
    task_opaque_ref>&uuid=<vm_uuid> -o vm.xva
<!--NeedCopy-->
```

Note that you can perform this operation on the pool coordinator even if the VM runs on a pool supporter. The request might result in a redirect if the VM's disks are placed on an SR accessible only by a pool member.

If you want to export only the VM metadata without the disks, use the HTTP(S) handler `export_metadata`. For example:

```
curl https://server/export_metadata?session_id=<session_opaque_ref>&ref
    =<vm_opaque_reference> -o vm_meta.xva
<!--NeedCopy-->
```

VM Import

To import a VM from an XVA package, construct an HTTP(S) PUT call. The call accepts the following query parameters:

Parameter	Description
`session_id`	The opaque reference of the session being used to authenticate.
`sr_id`	(Optional) The opaque reference of the SR on which the imported VM's disks will be placed. If not specified, the disks will be imported to `Pool.default_SR`. If the latter if not set, the error `DEFAULT_SR_NOT_FOUND` is returned.
`restore`	(Optional) If **`true`**, the import is treated as replacing the original VM from which the XVA package was exported. The implication of this currently is that the MAC addresses on the VIFs are exactly as the export was, which will lead to conflicts if the original VM is still running.
`force`	(Optional) If **`true`**, any checksum failures will be ignored (the default is to destroy the VM if a checksum error is detected).

Parameter	Description
`task_id`	(Optional) The opaque reference of a task object with which to monitor the progress of the operation; the task object needs to be created first

For example, to import a VM from package `vm.xva` placing its disks on a specific SR:

```
curl -T vm.xva http://server/import?session_id=<session_opaque_ref>&
    sr_id=<sr_opaque_ref>
<!--NeedCopy-->
```

Or, to import a VM from package `vm.xva` placing its disks on the default SR, while monitoring the progress of the operation:

```
curl -T vm.xva http://server/import?session_id=<session_opaque_ref>&
    task_id=<task_opaque_ref>
<!--NeedCopy-->
```

To import just the metadata, use the HTTP(S) handler `import_metadata`:

```
curl -T vm.xva http://server/import_metadata?session_id=<
    session_opaque_ref>
<!--NeedCopy-->
```

Xen Virtual Appliance (XVA) VM Import Format

XenServer supports a human-readable VM input format called XVA. This section describes the syntax and structure of the format.

An XVA is essentially a `tar` archive containing XML metadata and a set of disk images. A VM represented by an XVA is not intended to be directly executable. The data within an XVA package is intended for either archiving on permanent storage or for being transmitted to a VM server - such as a XenServer host - where it can be extracted and run.

XVA is a hypervisor-neutral packaging format; it is possible to create simple tools to instantiate an XVA VM on any other platform. XVA does not specify any particular runtime format; for example disks may be instantiated as file images, LVM volumes, QCoW images, VMDK or VHD images. An XVA VM may be instantiated any number of times, each instantiation may have a different runtime format.

Unlike other formats like the Open Virtual Appliance (OVA) format, XVA does not:

- specify any particular serialization or transport format
- provide any mechanism for customizing VMs (or templates) on install
- address how a VM may be upgraded post-install
- define how multiple VMs, acting as an appliance, may communicate.

The extracted content of an XVA file is a directory containing, at a minimum, a file called `ova.xml`. This file contains the metadata describing the VM packaged within the XVA. Each disk image is stored within a sub-directory and is referenced from the `ova.xml`. For example, the XVA describing a VM with two disks would contain the following metadata file and sub-directories:

```
$ls -al
total 72
drwx------+ 1 user1 Domain Users     0 Nov  7 10:35 .
drwxrwx---+ 1 user1 Domain Users     0 Nov  7 10:35 ..
-r-x------+ 1 user1 Domain Users 56046 Jan  1  1970 ova.xml
drwx------+ 1 user1 Domain Users     0 Nov  7 10:34 Ref_6
drwx------+ 1 user1 Domain Users     0 Nov  7 10:34 Ref_9
<!--NeedCopy-->
```

The file `ova.xml` contains the VM metadata in XML-RPC format, for example:

```
<value>
  <struct>
    <member>
      <name>objects</name>
      <value>
        <array>
          <data>
            <value>
              <struct>
                <member><name>class</name><value>VM</value></member>
                <member><name>id</name><value>Ref:VM</value></member>
                <member>
                  <name>snapshot</name>
                  <value>
                    <struct>
                      <member><name>power_state</name><value>Halted</
                          value></member>
                      <member><name>name_label</name><value>Test VM</
                          value></member>
                      <member><name>memory_static_max</name><value>4294967296</value></member>
                      <member><name>VCPUs_max</name><value>2</value></member>
                      <member><name>VIFs</name><value><array><data><value>Ref:VIF</value></data></array></value></member>
                      <member><name>PV_bootloader</name><value>pygrub</value></member>
                      <member><name>HVM_boot_policy</name><value>BIOS order</value></member>
                      <!-- ... -->
                    </struct>
                  </value>
                </member>
              </struct>
            </value>
            <value>
              <struct>
                <member><name>class</name><value>VIF</value></member>
                <member><name>id</name><value>Ref:VIF</value></member>
                <member>
                  <name>snapshot</name>
                  <value>
                    <!--...-->
                  </value>
                </member>
              </struct>
            </value>
            <!--...-->
          </data>
        </array>
      </value>
    </member>
  </struct>
</value>
<!--NeedCopy-->
```

A single disk image is represented by a directory containing a sequence of files as follows:

```
$ls -al Ref_6
total 2054
drwx------+ 1 user1 Domain Users       0 Nov  7 13:40 .
drwx------+ 1 user1 Domain Users       0 Nov  7 10:35 ..
-r-x------+ 1 user1 Domain Users 1048576 Jan  1  1970 00000000
-r-x------+ 1 user1 Domain Users      16 Jan  1  1970 00000000.xxhash
-r-x------+ 1 user1 Domain Users 1048576 Jan  1  1970 00000499
-r-x------+ 1 user1 Domain Users      16 Jan  1  1970 00000499.xxhash
...
<!--NeedCopy-->
```

Each of the extensionless files contains a block of raw disk image data of size 1MiB. The filename is the block number in decimal. The small size was chosen to be safely under the maximum file size limit of several filesystems. If these files are concatenated, the original image is recovered.

Each of the data block files is accompanied by a file with the same base name and the extension `.xxhash`, which contains the checksum of the corresponding data block file calculated using the xxHash algorithm.

> **Note:**
>
> In XenServer versions prior to Citrix Hypervisor 8.1, the checksums of the disk image blocks were calculated using the SHA1 algorithm, and the files containing the checksums had the extension `.checksum`. For example, a directory representing a disk image would contain the following sequence of files:
>
> ```
> $ls -al
> total 3079
> drwx------+ 1 user1 Domain Users       0 Nov  7 14:20 .
> drwx------+ 1 user1 Domain Users       0 Nov  7 10:35 ..
> -rwx------+ 1 user1 Domain Users 1048576 Aug  4 09:50
>     00000000000000000000
> -rwx------+ 1 user1 Domain Users      40 Aug  4 09:50
>     00000000000000000000.checksum
> -rwx------+ 1 user1 Domain Users 1048576 Aug  4 09:50
>     00000000000000000001
> -rwx------+ 1 user1 Domain Users      40 Aug  4 09:50
>     00000000000000000001.checksum
> -rwx------+ 1 user1 Domain Users 1048576 Aug  4 09:50
>     00000000000000000003
> -rwx------+ 1 user1 Domain Users      40 Aug  4 09:50
>     00000000000000000003.checksum
> ...
> <!--NeedCopy-->
> ```

Getting XenServer Operational Metrics

XenServer records metrics about the operation of various aspects of your XenServer installation. The metrics are stored persistently for long term access and analysis of historical trends.

Metrics available

A wide range of metrics are available for XenServer including: C-State, P-State, IOPS, Latency and many more. Not all of these are turned on by default. The full range of metrics available for both Hosts and VMs are detailed in section
Monitor XenServer Performance
of XenServer's documentation. This chapter also details how you can explore these metrics via XenCenter.

The data store for operational metrics

Metrics are stored in Round Robin Databases (RRDs), which are maintained for individual VMs (including the control domain) and the server. Each database consists of multiple Round Robin Archives (RRAs) and is of a fixed size. This is because each RRA is in effect a circular buffer with a predefined maximum capacity.

The VM RRDs are resident on the server on which the VM is running, or the pool coordinator when the VM is not running. This means that the location of the VM must be known in order to retrieve the associated data.

The RRDs are also backed up every day. Metrics are persisted for a maximum of one year, and are stored at different granularities.

RRDs are saved to disk as uncompressed XML. The size of each RRD when written to disk ranges from 200KiB to approximately 1.2MiB when the RRD stores the full year of metrics.

> **Note:**
>
> If metrics cannot be written to disk, for example when a disk is full, metrics will be lost and the last saved version of the RRD will be used.

Data Granularity

Each archive in the database samples its particular metric at a specified granularity. The values are stored at intervals of:

- 5 seconds for the past 10 minutes
- one minute for the past 2 hours

- one hour for the past week
- one day for the past year

The sampling that takes place every 5 seconds records actual data points, however, the following RRAs contain Consolidated Data Points (CDPs) which are produced by applying the Consolidation Functions (CFs) to a number of actual data points, storing historical data in a far more compressed format. The CFs supported by XenServer are:

- AVERAGE
- MIN
- MAX

Downloading RRDs

Metrics can be downloaded over HTTP(s), for example using `wget`.

> **Note:**
>
> In early versions of the XenServer Management API, instantaneous operational metrics could be obtained using the `VM_metrics`, `VM_guest_metrics`, `host_metrics` classes and associated methods. These methods have been deprecated in favor of using the HTTP(S) interface described in this chapter to download the metrics from the RRDs on the VMs and servers. The legacy metrics APIs will not return any values.

Metrics can be downloaded in `xport` style XML or JSON format. See rrddump and rrdxport for information about the XML format. The JSON format has the same structure as the XML.

Starting from xapi version 23.17.0, the server decides which format to return using the HTTP header `Accept`. To download metrics in XML format use `'Accept: text/xml'`, while to download metircs in JSON format use `'Accept: application/json'`. When both formats are accepted, for example, when using `'Accept: */*'` (which is the default for the `wget` client), the server returns JSON. The content type is provided in the reponse's headers.

> **Note:**
>
> When no header is specified, for example, when using `'Accept: '`, the server returns XML; to retrieve JSON you need to use the query parameter `json`. The header takes precedence over the query parameter and is the recommended method for deciding the returned format.

Downloading the whole RRD RRDs can be downloaded from the XenServer host on which they reside by
using the HTTP(S) handler registered at `/host_rrd` or `/vm_rrd`.

For example, to download a host RRD:

```
wget https://<server>/host_rrd?session_id=<session_opaque_reference>
<!--NeedCopy-->
```

To download a VM RRD, you must also specify the VM's `uuid` as a query
parameter:

```
wget https://<server>/vm_rrd?session_id=<session_opaque_reference>&uuid
   =<vm_uuid>
<!--NeedCopy-->
```

Getting updates from the RRD So as to not have to download the whole database when you have most of
the data already, a query can also be made to retrieve just updates.
This is achieved via the HTTP(S) handler `/rrd_updates`.

Updates are described in relation to a start time which is specified as
epoch time (number of seconds since Jan 1st, 1970) in a query parameter
named `start`.

To obtain an update of all VM metrics on a server, the URL would be
of the form:

```
wget http://<server>/rrd_updates?session_id=<session_opaque_reference>&
   start=<start_time>
<!--NeedCopy-->
```

The response contains data for every VM resident on the particular host
that is being queried. This means that it is not possible to query a
particular VM on its own. To differentiate which column in the export is
associated with which VM, the `legend` field is prefixed with the VM's
`uuid`. The data set also contains a prefix describing the CF used to
collect the data (`MAX`/`MIN`/`AVERAGE`).

To obtain host updates too, add the query parameter `host=true`:

```
wget http://<server>/rrd_updates?session_id=<session_opaque_reference>&
    start=<start_time>&host=true
<!--NeedCopy-->
```

When using RRD updates, you should be aware of the following:

- `/rrd_updates` will only return data for the metrics that are currently being collected. For example, if you have an unplugged VBD, `/vm_rrd` will return the historical data for that VBD from when it was in use, but `/rrd_updates` won't.
- The value of the `start` parameter is used to deduce the granularity of the data `/rrd_updates` will return. If you specify a shorter time period, you will get more detailed metrics. For example, if you specify a `start` value that is 9 minutes before "now", you'll get 108 rows from the 10 minute RRA, but if you specify 11 minutes before "now", you'll get 11 rows from the 2 hour RRA.
- Of particular importance is to use the same definition of "now"as the server is using, since the server is likely to be in UTC and your client may be in a different time zone.
- The data returned by each call to `/rrd_updates` will contain a value (in the `end` XML tag or JSON field) that can be used as the `start` parameter for the next call.

Additional `/rrd_updates` parameters

Parameter	Value	Description
`cf`	`average|min|max`	the data consolidation mode
`interval`	`<interval>`	the interval between values to be reported

Note:

By default only `average` metrics are available. To obtain `min` and `max` metrics for a VM, run the following command:

```
xe pool-param-set uuid=<pool_uuid> other-config:
    create_min_max_in_new_VM_RRDs
<!--NeedCopy-->
```

Analysing RRDs

Once you have obtained the requested RRDs, you can use a utility such as
rrdtool that will
allow you, amongst other forms of analysis, to plot graphs very easily,
or you can parse the results yourself.

This might be useful in if you want to perform simple tasks such as
determining what the current `Network` throughput is, or perhaps
carrying out your own, more complex analysis on the data points.

Some examples written in Python to demonstrate how you can parse and use
the metrics can be found at
XenServer Operational Metrics.

Example RRDs

Whole RRDs Below is an example XML for a whole host RRD. The table highlights
certain XML tags of interest.

XML tag	Description
`ds`	A data source field.
`name`	The name of this data source.
`type`	Defines the data source type. Can be `GAUGE`, `COUNTER`, `DERIVE` or `ABSOLUTE`.
`minimal_heartbeat`	Defines the maximum number of seconds before a `ds` value is considered unknown.
`min`	The minimum acceptable value. Values less than this number are considered unknown. This is optional.
`max`	The maximum acceptable value. Values exceeding this number are considered unknown. This is optional.

```xml
<?xml version="1.0"?>
<rrd>
  <version>0003</version>
  <step>5</step>
  <lastupdate>1213616574</lastupdate>
  <ds>
    <name>memory_total_kib</name>
```

```
      <type>GAUGE</type>
      <minimal_heartbeat>300.0000</minimal_heartbeat>
      <min>0.0</min>
      <max>Infinity</max>
      <last_ds>2070172</last_ds>
      <value>9631315.6300</value>
      <unknown_sec>0</unknown_sec>
    </ds>
    <ds>
     <!-- other data sources - the order of the data sources is important
          and defines the ordering of the columns in the archives below
              -->
    </ds>
    <rra>
      <cf>AVERAGE</cf>
      <pdp_per_row>1</pdp_per_row>
      <params>
        <xff>0.5000</xff>
      </params>
      <cdp_prep> <!-- This is for internal use -->
        <ds>
          <primary_value>0.0</primary_value>
          <secondary_value>0.0</secondary_value>
          <value>0.0</value>
          <unknown_datapoints>0</unknown_datapoints>
        </ds>
        ...other data sources - internal use only...
      </cdp_prep>
      <database>
       <row>
          <v>2070172.0000</v>  <!-- columns correspond to the DSs defined
               above -->
          <v>1756408.0000</v>
          <v>0.0</v>
          <v>0.0</v>
          <v>732.2130</v>
          <v>0.0</v>
          <v>782.9186</v>
          <v>0.0</v>
          <v>647.0431</v>
          <v>0.0</v>
          <v>0.0001</v>
          <v>0.0268</v>
          <v>0.0100</v>
          <v>0.0</v>
          <v>615.1072</v>
       </row>
       ...
    </rra>
    ... other archives ...
</rrd>
<!--NeedCopy-->
```

Here is an example JSON for a whole host RRD:

```
{

    "version": "0003",
    "step": "5",
    "lastupdate": "1698240426.8141",
    "ds": [
        {

            "name": "memory_total_kib",
            "type": "GAUGE",
            "minimal_heartbeat": "300.0000",
            "min": "0.0",
            "max": "Infinity",
            "last_ds": "33371836",
            "value": "60540111.1548",
            "unknown_sec": "0"
         }

        ... other data sources...
    ],
    "rra": [
        {

            "cf": "AVERAGE",
            "pdp_per_row": "1",
            "params": {

                "xff": "0.5000"
             }
 ,
            "cdp_prep": {

                "ds": [
                    {

                        "primary_value": "0.00",
                        "secondary_value": "0.00",
                        "value": "0.0",
                        "unknown_datapoints": "0.00"
                     }

                    ... other data sources - internal use only...
                ]
             }
 ,
            "database": [
                [
                    "0.0084",
                    "0.0090",
                    "0.0113",
                    "0.0072",
                    "0.0104",
                            "0.0099",
                            "0.0112",
                            "0.0131",
                            "0.0101",
                            "0.0100",
                            ... other values for the first data source...
                        ],
                        ... other data sources...
                    ]
                }
    ,
            ... other archives...
        ]
}

<!--NeedCopy-->
```

Note that a VM RRD is identically structured, but with different data sources.

RRD updates Example `rrd_updates` XML for one VM and no host updates:

```
<xport>
  <meta>
    <start>1213578000</start>
    <step>3600</step>
    <end>1213617600</end>
    <rows>12</rows>
    <columns>12</columns>
    <legend>
      <entry>AVERAGE:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu1</
          entry> <!-- nb - each data source might have multiple entries
          for different consolidation functions -->
      <entry>AVERAGE:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu0</
          entry>
      <entry>AVERAGE:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:memory</
          entry>
      <entry>MIN:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu1</entry>
      <entry>MIN:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu0</entry>
      <entry>MIN:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:memory</entry>
      <entry>MAX:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu1</entry>
      <entry>MAX:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu0</entry>
      <entry>MAX:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:memory</entry>
      <entry>LAST:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu1</entry>
      <entry>LAST:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:cpu0</entry>
      <entry>LAST:vm:ecd8d7a0-1be3-4d91-bd0e-4888c0e30ab3:memory</entry
          >
    </legend>
  </meta>
  <data>
    <row>
      <t>1213617600</t>
```

```
        <v>0.0</v> <!-- once again, the order or the columns is defined
           by the legend above -->
        <v>0.0282</v>
        <v>209715200.0000</v>
        <v>0.0</v>
        <v>0.0201</v>
        <v>209715200.0000</v>
        <v>0.0</v>
        <v>0.0445</v>
        <v>209715200.0000</v>
        <v>0.0</v>
        <v>0.0243</v>
        <v>209715200.0000</v>
      </row>
     ...
   </data>
</xport>
<!--NeedCopy-->
```

Example `rrd_updates` JSON for one VM and no host updates:

```
{

    "meta": {

        "start": "1213660800",
        "step": "86400",
        "end": "1698192000",
        "rows": "366",
        "columns": "101",
        "legend": [
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu0",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu1",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu2",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu3",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu4",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu5",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu6",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:cpu7",
            "AVERAGE:vm:b36c4883-7dca-41c6-9f35-ec91826ec29f:memory",
            ...
        ]
     }
  ,
    "data": [
        {

            "t": "1698192000",
            "values": [
                "0.0082",
                "0.0080",
                "0.0072",
                "0.0072",
                "0.0069",
                "0.0073",
                "0.0069",
                "0.0076",
                "2724740352.0000",
                "NaN",
                "NaN",
                ...
            ]
        }
  ,
        ...
    ]
  }

  <!--NeedCopy-->
```

Using the XenServer PowerShell module

March 19, 2024

This article describes how to use the XenServer PowerShell module to manage XenServer resource pools. The article includes an overview of the module cmdlets and their structure, followed by a number of walk-throughs on how to perform certain specialized tasks.

Getting Started

For information about the system requirements for the PowerShell module and how to download and install it, see Getting Started - PowerShell.

XenServer PowerShell module overview

XenServer sessions

The first cmdlet you will need is `Connect-XenServer` to open a session to a server:

```
Connect-XenServer -Url https://<servername> -UserName user -Password
    pwd
<!--NeedCopy-->
```

It is possible to connect to multiple servers using the same credentials as well
as open more than one sessions to the same server. All open sessions can be listed

using the cmdlet `Get-XenSession`. If more than one sessions are to be used, you can set the most frequently used session as the default one as follows:

```
Connect-XenServer -Server srv -UserName usr -Password pwd -
    SetDefaultSession
<!--NeedCopy-->
```

or

```
$XenServer_Default_Session = Get-XenSession -Server srv
<!--NeedCopy-->
```

All XenAPI calls are made in the context of a login session, so all cmdlets accept the parameter `-SessionOpaqueRef` which allows you to specify which of the open XenServer sessions to use:

```
Verb-Noun [-SessionOpaqueRef [<String>]]
<!--NeedCopy-->
```

This parameter is not necessary when only one open session exists or when a default session has been specified.

Once you have finished interacting with a server, it is good practice to log out using the cmdlet `Disconnect-XenServer`:

```
Get-XenSession | Disconnect-XenServer
<!--NeedCopy-->
```

Managing XenAPI objects

The cmdlets fall into the following categories:

1. Class getters These retrieve a XenAPI object and have names such as `Get-XenT`, where `T` is an exposed XenAPI class. The object to get can be specified by `-Ref` or, for those that have a uuid or name, `-Name` or `-Uuid`. If no parameters are specified, all objects of this type are returned. Example:

```
Get-XenHost -Name "Demo Host"
<!--NeedCopy-->
```

2. Constructors These create a new XenAPI object and have names such as `New-XenT`, where `T` is an exposed XenAPI class. Example:

```
$vm = Get-XenVM -Name "Demo VM"
New-XenVBD -VM $vm -VDI $null -Userdevice 3 -Bootable $false -Mode RO `
```

```
    -Type CD -Unpluggable $true -Empty $true -OtherConfig @{
  }
  `
    -QosAlgorithmType "" -QosAlgorithmParams @{
  }

<!--NeedCopy-->
```

3. Class removers These destroy a XenAPI object and have names such as `Remove-XenT`, where `T` is
an exposed XenAPI class. To specify the object to remove use the parameter
`-T`, where `T` is the exposed XenAPI class, or `-Ref` or, for those objects that
have a uuid or name, `-UUID` or `-Name`. Example:

```
Get-XenSR -Name "Demo SR" | Remove-XenSR
<!--NeedCopy-->
```

4. Property setters These set a field of a XenAPI object and have names such as `Set-XenT`, where
`T` is an exposed XenAPI class. To specify the object use the parameter `-T`, where
`T` is the exposed XenAPI class, or `-Ref` or, for those objects that have a uuid
or name, `-UUID` or `-Name`. The field to set can be specified as an accordingly
named parameter. Note that more than one fields at a time can be set in a
synchronous call. Example:

```
Get-XenVM -Name "Demo VM" |`
    Set-XenVM -NameLabel "New name" -NameDescription "New description"
<!--NeedCopy-->
```

5. Property adders These add an element to a field of a XenAPI object and have names such as
`Add-XenT`, where `T` is an exposed XenAPI class. To specify the object use the
parameter `-T`, where `T` is the exposed XenAPI class, or `-Ref` or, for those objects
that have a uuid or name, `-UUID` or `-Name`. The field to which the element
will be added can be specified as an accordingly named parameter. Note that
elements can be added to more than one fields at a time in a synchronous call.
Example:

```
Add-XenHost -Name "Demo Host" -Tags "Tag1"
<!--NeedCopy-->
```

6. Property removers These remove an element from a field of a XenAPI object and have names
such

as Remove-XenTProperty, where T is an exposed XenAPI class. To specify the object use the parameter -T, where T is the exposed XenAPI class, or -Ref or, for those objects that have a uuid or name, -UUID or -Name. The field from which the element will be removed can be specified as an accordingly named parameter. Note that elements can be removed from more than one fields at a time in a synchronous call. Example:

```
Remove-XenHostProperty -Name "myHost" -Tags "tag1" -OtherConfig "myKey"
<!--NeedCopy-->
```

7. Property getters These retrieve the value of a field of a XenAPI object and have names such as Get-XenTProperty, where T is an exposed XenAPI class. To specify the object use the parameters -T, where T is the exposed XenAPI class, or -Ref. To specify the field use the enum parameter -XenProperty. Example:

```
Get-XenPIFProperty -Ref OpaqueRef:f433bf7b-2b0c-5f53-7018-7d195addb3ca
    `
    -XenProperty Network
<!--NeedCopy-->
```

8. Invokers These invoke operations on XenAPI objects and have names such as Invoke-XenT, where T is an exposed XenAPI class. To specify the object use the parameter -T, where T is the exposed XenAPI class, or -Ref or, for those objects that have a uuid or name, -UUID or -Name. To specify the call to invoke, use the enum parameter -XenAction. Example:

```
Get-XenPBD -Uuid 1871ac51-ce6b-efc3-7fd0-28bc65aa39ff |`
    Invoke-XenPBD -XenAction Unplug
<!--NeedCopy-->
```

Most of the XenAPI calls can be run synchronously or asynchronously. To run a cmdlet asynchronously use the parameter -Async where available.

Note that, in the case of the setters, only one field can be set asynchronously at a time, and in the case of the adders and removers, elements can be added or removed asynchronously to only one field at a time.

Also, note that the cmdlets that are not explicit "getters"but return objects, do so only when the standard parameter -PassThru is specified. These cmdlets are Connect-XenServer, the constructors, adders, setters, certain invokers, the property removers, as well as all the asynchronous calls from any category. In the latter case, the cmdlet returns a Task object, the progress of which can be tracked by piping it into the cmdlet Wait-XenTask:

```
Invoke-XenVM -Name $vm_name -XenAction Start -Async -PassThru |`
    Wait-XenTask -ShowProgress
<!--NeedCopy-->
```

`Wait-XenTask`, too, can be used with the `-PassThru` parameter and, where applicable, it returns the opaque reference of the object that would be returned if the call were run synchronously.

Finally, as many of the cmdlets handle objects of type `XenRef<T>`, where `T` is an exposed API class, the cmldet `ConvertTo-XenRef` can be used to aid conversion between the two. Example:

```
Get-XenVM -Name "Demo VM" | ConvertTo-XenRef
<!--NeedCopy-->
```

Complete walk-throughs

How to configure vGPU and GPU pass-through

The following example is a tutorial of a typical vGPU configuration use case using the XenServer PowerShell SDK cmdlets. The example output provided is from a server with both a NVIDIA Grid K1 and K2 card installed.

We start by connecting to the server:

```
Connect-XenServer -Server server -UserName username -Password password
<!--NeedCopy-->
```

The first thing we probably want to check is which GPU groups exist (these have been created automatically once the graphics hardware is installed):

```
PS> Get-XenGPUGroup | select name_label, GPU_types,
   allocation_algorithm

name_label                                             GPU_types
   allocation_algorithm
----------                                             ---------
   --------------------
Group of NVIDIA Corporation GK104GL [GRID K2] GPUs   {
 10de/11bf }
             depth_first
Group of NVIDIA Corporation GK107GL [GRID K1] GPUs   {
 10de/0ff2 }
           breadth_first
<!--NeedCopy-->
```

The `allocation_algorithm` is the placement policy for assigning VMs to GPUs in order to achieve either maximum density by placing as many VMs as possible on the same GPU (`depth_first`), or maximum performance by placing VMs on as many GPUs as possible (`breadth_first`). For example, the following command sets the `allocation_algorithm` to achieve maximum performance:

```
PS> Get-XenGPUGroup | Set-XenGPUGroup -AllocationAlgorithm breadth_first
PS> Get-XenGPUGroup | select name_label, GPU_types, allocation_algorithm

name_label                                             GPU_types    allocation_algorithm
----------                                             ---------    --------------------
Group of NVIDIA Corporation GK104GL [GRID K2] GPUs   {
 10de/11bf }
           breadth_first
Group of NVIDIA Corporation GK107GL [GRID K1] GPUs   {
 10de/0ff2 }
           breadth_first
<!--NeedCopy-->
```

Now we can list some information about the vGPU types. The vGPU types are pre-sets which can be used to create different kinds of vGPUs.

```
PS> Get-XenVGPUType | ft vendor_name, model_name, framebuffer_size, max_heads, `
     max_resolution_x, max_resolution_y

vendor_name           model_name    framebuffer_size    max_heads    max_resolution_x    max_resolution_y
-----------           ----------    ----------------    ---------    ----------------    ----------------
NVIDIA Corporation    GRID K100            268435456            2                1920                1200
NVIDIA Corporation    GRID K140Q          1006632960            2                2560                1600
NVIDIA Corporation    GRID K240Q          1006632960            2                2560                1600
NVIDIA Corporation    GRID K260Q          2013265920            4                2560                1600
NVIDIA Corporation    GRID K200            268435456            2                1920                1200
                      passthrough                  0            0                   0                   0
<!--NeedCopy-->
```

The vGPU type `passthrough` is supported for all PCI display devices, and can be used to create pass-through vGPUs.

We can see for which PCI display devices a vGPU type is supported as follows:

```
PS> Get-XenVGPUType | Where {
 $_.model_name -eq "GRID K100" }
  |`
    Get-XenVGPUTypeProperty -XenProperty SupportedOnPGPUs |`
    Get-XenPGPU | Get-XenPCI -Ref {
 $_.PCI }
  | select device_name, pci_id

device_name          pci_id
-----------          ------
GK107GL [GRID K1]    0000:0a:00.0
GK107GL [GRID K1]    0000:07:00.0
GK107GL [GRID K1]    0000:08:00.0
GK107GL [GRID K1]    0000:09:00.0
<!--NeedCopy-->
```

A vGPU type will show up as supported or enabled in a GPU group if it is supported or enabled respectively on at least one of the pGPUs in the group. We can query the GPU groups to find out which vGPU types are supported or enabled. For example:

```
PS> Get-XenGPUGroup | Where {
 $_.name_label -match "GRID K2" }
  |`
    Get-XenGPUGroupProperty -XenProperty EnabledVGPUTypes |`
    Get-XenVGPUType | select model_name

model_name
----------
GRID K200
passthrough
GRID K260Q
GRID K240Q
<!--NeedCopy-->
```

We may want to disallow a certain vGPU type on a pGPU. For example, the following commands disable and then re-enable the vGPU type Grid K240Q:

```
PS> $vgpuType = Get-XenVgpuType | where {
 $_.model_name -eq "GRID K240Q" }

PS> $vgpuType | Get-XenVGPUTypeProperty -XenProperty EnabledOnPGPUs |`
    Get-XenPGPU | select uuid

uuid
----
a913a90d-0be9-aea3-862f-3fbfe1823a3f
8a0d1316-0f09-1ee9-f95b-c778c759ee40

PS> Remove-XenPGPUProperty -Uuid a913a90d-0be9-aea3-862f-3fbfe1823a3f `
    -EnabledVGPUTypes $vgputype.opaque_ref

PS> $vgpuType | Get-XenVGPUTypeProperty -XenProperty EnabledOnPGPUs |
   get-xenpgpu | select uuid
```

```

uuid
----
8a0d1316-0f09-1ee9-f95b-c778c759ee40

PS> Add-XenPGPU -Uuid a913a90d-0be9-aea3-862f-3fbfe1823a3f -
    EnabledVGPUTypes $vgputype.opaque_ref
PS> $vgpuType | Get-XenVGPUTypeProperty -XenProperty EnabledOnPGPUs |
    get-xenpgpu | select uuid

uuid
----
a913a90d-0be9-aea3-862f-3fbfe1823a3f
8a0d1316-0f09-1ee9-f95b-c778c759ee40
<!--NeedCopy-->
```

We may want to find out how many vGPUs of a certain type can be started on the pGPUs in a group, in addition to the vGPUs which are already running:

```
PS> $gpuGroups = Get-XenGPUGroup
PS> Get-XenVGPUtype | % {
 Get-XenGPUGroupProperty $gpuGroups[0] -XenProperty RemainingCapacity -
    VgpuType $_ }


0
0
8
4
16
2
<!--NeedCopy-->
```

Now suppose we want to assign a vGPU of type Grid K260Q to a VM which has no vGPU yet. This can be done as follows:

```
PS> $vm = Get-XenVm -Name "w7-test"
PS> $vgpuTypes = Get-XenVGPUtype
PS> Get-XenVMProperty -VM $vm -XenProperty VGPUs
PS>
PS> New-XenVGPU -VM $vm -GPUGroup $gpuGroups[0] -Device 0 -Type
    $vgpuTypes[3] -PassThru

uuid                : f1122947-3b11-3fd3-0630-779701b37265
VM                  : XenAPI.XenRef`1[XenAPI.VM]
GPU_group           : XenAPI.XenRef`1[XenAPI.GPU_group]
device              : 0
currently_attached  : False
other_config        : {
  }

type                : XenAPI.XenRef`1[XenAPI.VGPU_type]
resident_on         : XenAPI.XenRef`1[XenAPI.PGPU]
```

```
opaque_ref              : OpaqueRef:15b2d1a9-8944-2f28-53df-6b8274d4d6fb
Changed                 : True
<!--NeedCopy-->
```

At this stage we can boot the VM, install the in-guest NVIDIA drivers, and enable RDP. Then we may want to disable the VNC console as this has a significant performance overhead. To do this we need to shut down the VM, set the flag `vgpu_vnc_enabled` to **`false`** and then boot the VM.

```
PS> Invoke-XenVM -VM $vm -XenAction Shutdown –Async –PassThru | Wait-
    XenTask –ShowProgress

PS> $p = Get-XenVMProperty -VM $vm -XenProperty Platform
PS> $p["vgpu_vnc_enabled"]="false"
PS> Set-XenVM -VM $vm -Platform $p
PS> Invoke-XenVM -VM $vm -XenAction Start –Async –PassThru | Wait-
    XenTask –ShowProgress
<!--NeedCopy-->
```

Before finishing we should remember to disconnect from the server:

```
Get-XenSession | Disconnect-XenServer
<!--NeedCopy-->
```

How to copy or live migrate a VM across pools

The following tutorial shows how to use the XenServer PS module to migrate a running VM or copy a halted VM from a pool or standalone server, hereafter referred to as the source server, to a different pool or standalone server, hereafter referred to as the destination server.

This example also displays how to work with multiple sessions and use implicitly the global variable `$XenServer_Default_Session`.

We're starting by connecting to the source server. The session created with the source server will be more frequently used, so it makes sense to set it as the default session:

```
Connect-XenServer -Url $sourceServerUrl -UserName $sourceServerUsername
     -Password $sourceServerPwd -SetDefaultSession
<!--NeedCopy-->
```

Let's select the VM we are going to copy or live migrate from the source pool:

```
$theVM = Get-XenVM -Name $nameOfSourceVmToCopy
<!--NeedCopy-->
```

Note that in the above call we did not need to specify the `–Session` parameter since we have set the source session as the default one.

Then we need to connect to the destination pool and obtain its coordinator. Note that we will need to specify the -Session parameter in all calls made to the destination pool since this session is not the default one.

```
$destSession = Connect-XenServer -Url $destServerUrl -UserName
    $destServerUsername -Password $destServerPwd -PassThru

$destCoordinator = Get-XenPool -Session $destSession.opaque_ref | `
     select -ExpandProperty master | `
     Get-XenHost -Session $destSession.opaque_ref
<!--NeedCopy-->
```

The coordinator of the destination pool has to be prepared to receive the VM that will be live migrated or copied. For this, we will also need to obtain the details of the network on the destination pool through which migration traffic will be received.

```
$destTransferNetwork = Get-XenNetwork -Name $destTransferNetworkName -
    Session $destSession.opaque_ref

$destMap = Invoke-XenHost -XenAction MigrateReceive -XenHost
    $destCoordinator -Network $destTransferNetwork -Session $destSession
    .opaque_ref -options @null -PassThru
<!--NeedCopy-->
```

The next step is to map the VM's VIFs to networks on the destination pool. Let's consider a case where the destination pool has as many networks as the VIFs on the VM we want to live migrate or copy, so we can create a one-to-one mapping of VIFs to networks as follows:

```
$targetNetworkRefs = Get-XenNetwork -SessionOpaqueRef $destSession.
    opaque_ref | ConvertTo-XenRef

$vmVifRefs = $theVM.VIFs | Get-XenVIF | ConvertTo-XenRef

$vifmap = @{
   }


for ($i = 0; $i -lt $vmVifRefs.Count; $i++) {

     $vifmap[$vmVifRefs[$i]] = $targetNetworkRefs[$i]
 }

<!--NeedCopy-->
```

Similarly, we proceed to map the VM's disks to storage repositories on the destination pool. Let's consider a simple case where we want to place all VM disks on the same storage repository with name $destSrName.

```
$targetSrRef = Get-XenSR -name $destSrName -SessionOpaqueRef
    $destSession.opaque_ref | ConvertTo-XenRef

$vdiRefs = $theVM.VBDs | Get-XenVBD | `
    where {
  $_.type -eq [XenAPI.vbd_type]::Disk }
  | `
    select -ExpandProperty VDI

$vdimap = @{
  }


foreach ($vdiRef in $vdiRefs) {

    $vdimap[$vdiRef] = $targetSrRef
 }

<!--NeedCopy-->
```

Now we are ready to live migrate or copy the VM to the destination pool. Both operations are performed using the same cmdlet. However, we need to specify different options in each case. The call is run on the source pool session. Due to the long duration of the operation, it is recommended to run it asynchronously.

To live migrate a running VM:

```
$theTask = Invoke-XenVM -VM $theVM -XenAction MigrateSend -Live $true `
    -Dest $destMap -VifMap $vifmap -VdiMap $vdimap `
    -Options @{
  }
  -Verbose -Async -PassThru
<!--NeedCopy-->
```

Or, to copy a halted VM:

```
$theTask = Invoke-XenVM -VM $theVM -XenAction MigrateSend -Live $false
    `

    -Dest $destMap -VifMap $vifmap -VdiMap $vdimap `
    -Options @{
 "copy"="true" }
  -Verbose -Async -PassThru
<!--NeedCopy-->
```

In either case, we can monitor the task progress:

```
Wait-XenTask -Task $theTask -ShowProgress
<!--NeedCopy-->
```

Once the task is finished, we can query it to obtain the new VM on the destination pool:

```
$theTask = Get-XenTask -uuid $theTask.uuid

if ($theTask.status -eq [XenAPI.task_status_type]::success) {

    $doc = [xml]$theTask.result
    $newVmRef = $doc.value
    Get-XenVM -Ref $newVmRef -SessionOpaqueRef $destSession.opaque_ref
 }

else {

    Write-Host "Operation failed: " $theTask.error_info
 }

<!--NeedCopy-->
```

Finally, we need to disconnect from the source and the destination pools:

```
Get-XenSession | Disconnect-XenServer
<!--NeedCopy-->
```

XenServer Management API extensions

November 27, 2023

The Management API is a general and comprehensive interface to managing the life-cycles of virtual machines, and offers a lot of flexibility in the way that Management API providers may implement specific functionality (for example, storage provisioning, or console handling). XenServer has several extensions which provide useful functionality used in our own XenCenter interface. The workings of these mechanisms are described in this chapter.

Extensions to the Management API are often provided by specifying `other-config` map keys to various objects. The use of this parameter indicates that the functionality is supported for that particular release of XenServer, but *not* as a long-term feature. We are constantly evaluating promoting functionality into the API, but this requires the nature of the interface to be well-understood. Developer feedback as to how you are using some of these extensions is always welcome to help us make these decisions.

VM console forwarding

Most Management API graphical interfaces will want to gain access to the VM consoles, in order to render them to the user as if they were physical machines. There are several types of consoles available, depending on the type of guest or if the physical host console is being accessed:

Console access

Operating System	Text	Graphical	Optimized graphical
Windows	No	VNC, using an API call	RDP, directly from guest
Linux	Yes, through VNC and an API call	No	VNC, directly from guest
Physical Host	Yes, through VNC and an API call	No	No

Hardware-assisted VMs, such as Windows, directly provide a graphical console over VNC. There is no text-based console, and guest networking is not necessary to use the graphical console. Once guest networking has been established, it is more efficient to setup Remote Desktop Access and use an RDP client to connect directly (this must be done outside of the Management API).

Paravirtual VMs, such as Linux guests, provide a native text console directly. XenServer provides a utility (called `vncterm`) to convert this text-based console into a graphical VNC representation. Guest networking is not necessary for this console to function. As with Windows above, Linux distributions often configure VNC within the guest, and directly connect to it over a guest network interface.

The physical host console is only available as a `vt100` console, which is exposed through the Management API as a VNC console by using `vncterm` in the control domain.

RFB (Remote Framebuffer) is the protocol which underlies VNC, specified in The RFB Protocol.
Third-party developers are expected to provide their own VNC viewers, and many freely available implementations can be adapted for this purpose. RFB 3.3 is the minimum version which viewers must support.

Retrieving VNC consoles using the API

VNC consoles are retrieved using a special URL passed through to the host agent. The sequence of API calls is as follows:

1. Client to Master/443: RPC: `Session.login_with_password()`.
2. Master/443 to Client: Returns a session reference to be used with subsequent calls.
3. Client to Master/443: RPC: `VM.get_by_name_label()`.
4. Master/443 to Client: Returns a reference to a particular VM (or the "control domain"if you want to retrieve the physical host console).
5. Client to Master/443: RPC: `VM.get_consoles()`.
6. Master/443 to Client: Returns a list of console objects associated with the VM.
7. Client to Master/443: RPC: `VM.get_location()`.
8. Returns a URI describing where the requested console is located. The URIs are of the form: `https://192.168.0.1/console?ref=OpaqueRef:c038533a-af99-a0ff-9095-c1159f2dc6a0` or `https://192.168.0.1/console?uuid=026e34fe-f0f2-20ee-5344-46d1aa922d5b`
9. Client to 192.168.0.1: HTTP CONNECT "/console?ref=(…)". You will also need to pass in "session_id=<session reference>"as a cookie.

The final HTTP CONNECT is slightly non-standard since the HTTP/1.1 RFC specifies that it must only be a host and a port, rather than a URL. Once the HTTP connect is complete, the connection can subsequently directly be used as a VNC server without any further HTTP protocol action.

This scheme requires direct access from the client to the control domain's IP, and will not work correctly if there are Network Address Translation (NAT) devices blocking such connectivity. You can use the CLI to retrieve the console URI from the client and perform a connectivity check.

Retrieve the VM UUID by running:

```
    xe vm-list params=uuid --minimal name-label=name
<!--NeedCopy-->
```

Retrieve the console information:

```
    xe console-list vm-uuid=uuid
    uuid ( RO): 714f388b-31ed-67cb-617b-0276e35155ef
    vm-uuid ( RO): 8acb7723-a5f0-5fc5-cd53-9f1e3a7d3069
    vm-name-label ( RO): etch
    protocol ( RO): RFB
    location ( RO): https://192.168.0.1/console?ref=(...)
<!--NeedCopy-->
```

Use command-line utilities like `ping` to test connectivity to the IP address provided in the `location` field.

Disabling VNC forwarding for Linux VM

When creating and destroying Linux VMs, the host agent automatically manages the `vncterm` processes which convert the text console into VNC. Advanced users who want to directly access the text console can disable VNC forwarding for that VM. The text console can then only be accessed directly from the control domain directly, and graphical interfaces such as XenCenter will not be able to render a console for that VM.

Before starting the guest, set the following parameter on the VM record:

```
xe vm-param-set uuid=uuid other-config:disable_pv_vnc=1
```

Start the VM.

Use the CLI to retrieve the underlying domain ID of the VM with:

```
xe vm-list params=dom-id uuid=uuid --minimal
```

On the host console, connect to the text console directly by:

```
/usr/lib/xen/bin/xenconsole domain_id
```

This configuration is an advanced procedure, and we do not recommend that the text console is directly used for heavy I/O operations. Instead, connect to the guest over SSH or some other network-based connection mechanism.

Adding xenstore entries to VMs

Developers might want to install guest agents into VMs which take special action based on the type of the VM. In order to communicate this information into the guest, a special xenstore name-space known as

vm-data is available which is populated at VM creation time. It is
populated from the xenstore-data map in the VM record.

Set the xenstore-data parameter in the VM record:

```
xe vm-param-set uuid=vm_uuid xenstore-data:vm-data/foo=bar
```

Start the VM.

If it is a Linux-based VM, install the XenServer VM Tools and use the command xenstore-read to verify that the node exists in xenstore.

> **Note:**
>
> Only prefixes beginning with vm-data are permitted, and anything not
> in this name-space will be silently ignored when starting the VM.

Security enhancements

The control domain in XenServer has
various security enhancements in order to harden it against attack from
malicious guests. These changes do not result in any loss of correct
functionality, but the changes are documented here
as variations of behavior from other distributions.

- The socket interface, xenstored, access using libxenstore. Interfaces are restricted by xs_restrict().
- The device /dev/xen/evtchn, which is accessed by calling
  xs_evtchn_open() in libxenctrl. A handle can be restricted using
  xs_evtchn_restrict().
- The device /proc/xen/privcmd, accessed through
  xs_interface_open() in libxenctrl. A handle is restricted using
  xc_interface_restrict(). Some privileged commands are naturally
  hard to restrict (for example, the ability to make arbitrary hypercalls),
  and these are simply prohibited on restricted handles.
- A restricted handle cannot later be granted more privilege, and so
  the interface must be closed and re-opened. Security is only gained
  if the process cannot subsequently open more handles.

The control domain privileged user-space interfaces can now be
restricted to only work for certain domains. There are three interfaces
affected by this change:

- The `qemu` device emulation processes and `vncterm` terminal emulation processes run as a non-root user ID
  and are restricted into an empty directory. They uses the
  restriction API above to drop privileges where possible.
- Access to xenstore is rate-limited to prevent malicious guests from
  causing a denial of service on the control domain. This is
  implemented as a token bucket with a restricted fill-rate, where
  most operations take one token and opening a transaction takes 20.
  The limits are set high enough that they are never hit when
  running even a large number of concurrent guests under loaded
  operation.
- The VNC guest consoles are bound only to the `localhost` interface,
  so that they are not exposed externally even if the control domain
  packet filter is disabled by user intervention.

Advanced settings for network interfaces

Virtual and physical network interfaces have some advanced settings that
can be configured using the `other-config` map parameter. There is a set of custom ethtool settings and some miscellaneous settings.

ethtool settings

Developers might want to configure custom ethtool settings for physical
and virtual network interfaces. This is accomplished with
`ethtool-<option>` keys in the `other-config` map parameter.

Key	Description	Valid settings
ethtool-rx	Specify if RX checksumming is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it
ethtool-tx	Specify if TX checksumming is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it
ethtool-sg	Specify if scatter-gather is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it

Key	Description	Valid settings
ethtool-tso	Specify if tcp segmentation offload is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it
ethtool-ufo	Specify if UDP fragmentation offload is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it
ethtool-gso	Specify if generic segmentation offload is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it
ethtool-autoneg	Specify if autonegotiation is enabled	`on` or **`true`** to enable the setting, `off` or **`false`** to disable it
ethtool-speed	Set the device speed in Mb/s	10, 100, or 1000
ethtool-duplex	Set full or half duplex mode	half or full

For example, to enable TX checksumming on a virtual NIC using the xe
CLI:

```
xe vif-param-set uuid=<VIF UUID> other-config:ethtool-tx="on"
```

or:

```
xe vif-param-set uuid=<VIF UUID> other-config:ethtool-tx="true"
```

To set the duplex setting on a physical NIC to half duplex using the xe
CLI:

```
xe vif-param-set uuid=<VIF UUID> other-config:ethtool-duplex="half"
```

Miscellaneous settings

You can also set a promiscuous mode on a VIF or PIF by setting the `promiscuous` key to `on`. For example, to enable promiscuous mode on a physical NIC using the xe CLI:

```
xe pif-param-set uuid=<PIF UUID> other-config:promiscuous="on"
```

or:

```
xe pif-param-set uuid=<PIF UUID> other-config:promiscuous="true"
```

The VIF and PIF objects have a `MTU` parameter that is read-only and provide the current setting of the maximum transmission unit for the interface. You can override the default maximum transmission unit of a physical or virtual NIC with the `mtu` key in the `other-config` map parameter. For example, to reset the MTU on a virtual NIC to use jumbo frames using the xe CLI:

```
xe vif-param-set uuid=<VIF UUID> other-config:mtu=9000
```

Note that changing the MTU of underlying interfaces is an advanced and experimental feature, and may lead to unexpected side-effects if you have varying MTUs across NICs in a single resource pool.

Internationalization for SR names

The SRs created at install time now have an `other_config` key indicating how their names may be internationalized.

`other_config["i18n-key"]` may be one of

- `local-hotplug-cd`
- `local-hotplug-disk`
- `local-storage`
- `XenServer-tools`

Additionally, `other_config["i18n-original-value-<field name>"]` gives the value of that field when the SR was created. If XenCenter sees a record where `SR.name_label` equals `other_config["i18n-original-value-name_label"]` (that is, the record has not changed since it was created during XenServer installation), then internationalization will be applied. In other words, XenCenter will disregard the current contents of that field, and instead use a value appropriate to the user's own language.

If you change `SR.name_label` for your own purpose, then it no longer is the same as `other_config["i18n-original-value-name_label"]`. Therefore, XenCenter does not apply internationalization, and instead preserves your given name.

Hiding objects from XenCenter

Networks, PIFs, and VMs can be hidden from XenCenter by adding the key `HideFromXenCenter=true` to the `other_config` parameter for the object. This capability is intended for ISVs who know what they are doing, not general use by everyday users. For example, you might want to hide certain VMs because they are cloned VMs that are not intended to be used directly by general users in your environment.

In XenCenter, hidden Networks, PIFs, and VMs can be made visible, using the View menu.

XenCenter API Extensions

November 27, 2023

The following section details the assumptions and API extensions that we have made, over and above the documented API. Extensions are encoded as particular key-value pairs in dictionaries such as `VM.other_config`.

Pool

Key	Semantics
pool.name_label	An empty name_label indicates that the pool is hidden on the tree view.
pool.rolling_upgrade_in_progress	Present if the pool is in the middle of a rolling upgrade.

Host

Key	Semantics
host.other_config[“iscsi_iqn”]	The host’s iSCSI IQN.
host.license_params[“expiry”]	The expiry date of the host’s license, in ISO 8601, UTC.
host.license_params[“sku_type”]	The host license type i.e. Server or Enterprise.

Key	Semantics
host.license_params[“restrict_pooling”]	Returns `true` if pooling is restricted by the host.
host.license_params[“restrict_connection”]	The number of connections that can be made from XenCenter is restricted.
host.license_params[“restrict_qos”]	Returns `true` if Quality of Service settings are enabled on the host.
host.license_params[“restrict_vlan”]	Returns `true` if creation of virtual networks is restricted on the host.
host.license_params[“restrict_pool_attached_storage”]	Returns `true` if the creation of shared storage is restricted on this host.
host.software_version[“product_version”]	Returns the host’s product version.
host.software_version[“build_number”]	Returns the host’s build number.
host.software_version[“xapi”]	Returns the host’s api revision number.
host.software_version[“package-linux”]	Returns “installed”if the Linux pack has been installed.
host.software_version[“oem_build_number”]	If the host is the OEM version, return its revision number.
host.logging[“syslog_destination”]	Gets or sets the destination for the XenServer system logger (null for local logging).
host.logging[“multipathing”]	“true”if storage multipathing is enabled on this host.
host.logging[“boot_time”]	A floating point Unix time giving the time that the host booted.
host.logging[“agent_start_time”]	A floating point Unix time giving the time that the control domain management daemon started.

VM

Key	Semantics
VM.other_config[“default_template”]	This template is one that was installed by . This is used to selectively hide these in the tree view, to use a different icon for them, and to disallow deletion.
VM.other_config[“xensource_internal”]	This template is special, such as the P2V server template. These are completely hidden by the UI.

Key	Semantics
VM.other_config[“install_distro”] == “rhlike”	This template is for RHEL 5, or CentOS equivalents. This is used to prompt for the Install Repository during install, including support for install from ISO / CD, and to modify NFS URLs to suit these installers.
VM.other_config[“install-repository”] == “cdrom”	Requests an install from a repository in the VM’s attached CD drive, rather than a URL.
VM.other_config[“auto_poweron”]	Gets or sets whether the VM starts when the server boots, “true”or “false”.
VM.other_config[“ignore_excessive_vcpus”]	Gets or sets to ignore XenCenter warning if a VM has more VCPUs than its host has physical CPUs, `true` to ignore.
VM.other_config[“HideFromXenCenter”]	Gets or sets whether XenCenter will show the VM in the treeview, “true”to hide.
VM.other_config[“import_task”]	Gets the import task that created this VM.
VM.HVM_boot_params[“order”]	Gets or sets the VM’s boot order on HVM VM’s only, for example, “CDN”will boot in the following order - First boot disk, CD drive, Network.
VM.VCPU_params[“weight”]	Gets or sets the IONice value for the VM’s VCPUs, ranges from 1 to 65536, 65536 being the highest.
VM.pool_migrate(…, options[‘live’])	`true` indicates live migration. XenCenter always uses this.
VM.other_config[“install-methods”]	A comma-separated list of install methods available for this template. Can include “cdrom”, “nfs”, “http”or “ftp”.
VM.other_config[“last_shutdown_time”]	The time that this VM was last shut down or rebooted, formatted as a UTC ISO8601 datetime.
VM.other_config[“p2v_source_machine”]	The source machine, if this VM was imported by a P2V process.
VM.other_config[“p2v_import_date”]	The date the VM was imported, if it was imported by a P2V process. Formatted as a UTC ISO8601 datetime.

SR

Key	Semantics
SR.other_config[“auto-scan”]	The SR will be automatically scanned for changes. Set on all SRs created by XenCenter.
SR.sm_config[“type”]	Set as type `cd` for SRs which are physical CD drives.

VDI

Key	Semantics
VDI.type	`user` instead of `system` is used to mean “do or do not allow deletion of the VDI through the GUI, if this disk is attached to a VM”. The intention here is to prevent you from corrupting a VM (uninstall it instead). `suspend` and `crashdump` record suspend and core dumps respectively. `ephemeral` is currently unused.
VDI.managed	All unmanaged VDIs are completely hidden in the UI. These are branch points in VHD chains, or unused LUN-per-VDI disks.
VDI.sm_config[“vmhint”]	The UUID of the VM that this VDI supports. This is set when VDIs are created through the user interface, to improve performance for certain storage backends.

VBD

Key	Semantics
VBD.other_config[“owner”]	If set, then this disk may be deleted when the VM is uninstalled.
VBD.other_config[“class”]	Set to an integer, corresponding to the Best Effort setting of `ionice`.

Network

Key	Semantics
network.other_config[“automatic”]	The New VM wizard will create a VIF connected to this network by default, if this key has any value other than `false`.
network.other_config[“import_task”]	Gets the import task that created this network.

VM_guest_metrics

Key	Semantics
PV_drivers_version[“major”]	Gets the major version of the XenServer VM Tools on the VM.
PV_drivers_version[“minor”]	Gets the minor version of the XenServer VM Tools on the VM.
PV_drivers_version[“micro”]	Gets the micro (build number) of the XenServer VM Tools on the VM.

Task

Key	Semantics
task.other_config[“object_creation”] == “complete”	For the task associated with a VM import, this flag will be set when all the objects (VMs, networks) have been created. This is useful in the import VM wizard for us to then go and re-map all the networks that need it.

XenServer Changed Block Tracking Guide

May 10, 2023

Changed block tracking provides a set of features and APIs that enable you to develop fast and space-efficient incremental backup solutions for XenServer.

Changed block tracking is available only to customers with XenServer Premium Edition.
If a customer without Premium Edition attempts to use an incremental backup solution for XenServer

that uses changed block tracking, they are prevented from enabling changed block tracking on new VDIs.
However, if the customer has existing VDIs with changed block tracking enabled, they can still perform other changed block tracking actions on these VDIs.

How changed block tracking works

When changed block tracking is enabled for a virtual disk image (VDI), any blocks that are changed in that VDI are recorded in a log file.
Every time the VDI is snapshotted, this log file can be used to identify the blocks that have changed since the VDI was last snapshotted.
This provides the capability to backup only those blocks that have changed.

After the changed blocks have been exported, the full VDI snapshots can now be changed into metadata-only snapshots by destroying the data associated with them and leaving only the changed block information.
These metadata only snapshots are linked both to the preceding metadata-only snapshot and to the following metadata-only snapshot.
This provides a chain of metadata that records the full history of changes to this VDI since changed block tracking was enabled.

The changed block tracking feature also takes advantage of network block device (NBD) capabilities to perform the export of data from the changed blocks.

Benefits of changed block tracking

Unlike some other incremental backup solutions, changed block tracking does not require that the customer keep a snapshot of the last known good state of a VDI available on the host or a storage repository (SR) to compare the current state to.
The customer needs less disk space because, instead of handling and storing large VDIs, with changed block tracking they instead can choose to store space-efficient metadata-only snapshot files.

Changed block tracking also saves the customer time as well as space.
Other backup solutions export a snapshot of the whole VDI every single time the VDI is backed up.
This is a time-consuming process and the customer has to pay that time cost every time they take a backup.
With changed block tracking enabled, the first back up exports a snapshot of the whole VDI.
However, subsequent backups only export the blocks in the VDI that have changed since the previous backup.
This decreases the time required to export the backup in proportion to the percentage of blocks that have changed.

For example, it can take around 10 hours to export a backup of a full 1 TB VDI.
If, after a week, 5% of the blocks in that VDI have changed, exporting the backup takes 5% of the time - 30 minutes.
A backup taken after a day has even fewer changed blocks and takes even less time to export.

The savings in time and space that changed block tracking provides makes it a preferable backup solution for customers using XenServer.
The simple API that XenServer exposes makes it easy for you to develop an incremental backup solution that delivers these benefits to the end user.
You can use this API through the language-agnostic remote procedure calls (RPCs) or take advantage of the language bindings provided for C, C#, Java, Python and PowerShell.

Getting started using changed block tracking

October 18, 2023

This section steps through the process of using changed block tracking to create incremental backups.

Before getting started with changed block tracking, we recommend that you read the XenServer Software Developer Kit Guide.
This document contains information to help you become familiar with developing for XenServer.

The examples provided in these steps use the Python binding for the Management API.

- For more information about the individual RPC calls, see the XenServer Management API
- For more detailed information about individual steps in this process, see the following chapters.

Full Python examples are provided on GitHub.

The NBD connection examples provided in these steps use the Linux nbd-client.
However, you can use any NBD client that supports the "fixed newstyle"version of the NBD protocol.
For more information, see the NBD protocol documentation.

> **Note:**
>
> If you are using the Linux upstream NBD client, a minimum version of 3.15 is required to support TLS.

Prerequisites

Before you start, set up or implement an NBD client at the backup location that supports the "fixed newstyle"version of the NBD protocol.

For more information, see Exporting the changed blocks using an NBD client.

Enable NBD connections on your network.
For more information, see Enabling NBD connections on XenServer.

Procedure

This procedure is broken down into three sections:

- **Setting up changed block tracking**
  Perform the steps in this section once, when you start using changed block tracking, to enable the changed block tracking capability and export a base snapshot that the incremental, changed block exported data is compared to.
- **Taking incremental backups**
  Perform the steps in this section every time you want to take an incremental back up of the changed blocks in a VDI.
- **Restoring a VDI from exported changed block data**
  Perform the steps in this section if you want to use your backed up data to restore a VDI to an earlier state.

Setting up changed block tracking

Before you can take incremental backups of a VDI using changed block tracking, you must first enable changed block tracking on the VDI and export a base snapshot.
To set up changed block tracking for a VDI, complete the following steps

1. Use the Management API to establish a XenAPI session on the XenServer host:

   ```
   import XenAPI
   import shutil
   import urllib3
   import requests

   session = XenAPI.xapi_local()
   session.xenapi.login_with_password("<user>", "<password>", "<
       version>", "<originator>")
   ```

2. **Optional**: If you intend to create a new VM and new VDIs to restore your backed up data to, you must also export your VM metadata.
   Ensure that you export a copy of the VM metadata any time your VM properties change.
   This can be done by using HTTPS or by using the command line.

```
session_id = session._session
url = ("https://%s/export_metadata?session_id=%s&uuid=%s"
        "&export_snapshots=false"
        % (<xs_host>, session_id, <vm_uuid>))

with requests.Session() as session:
    urllib3.disable_warnings(urllib3.exceptions.
        InsecureRequestWarning)
    request = session.get(url, verify=False, stream=True)
    with open(<export_path>, 'wb') as filehandle:
        shutil.copyfileobj(request.raw, filehandle)
    request.raise_for_status()
```

Where <*export_path*> is the location to save the VM metadata to.

The export URL includes the parameter `export_snapshots=false`.
This parameter ensures that the snapshot history is not included in the VM metadata backup. The VM metadata is used to create a new VM and this snapshot history does not apply to the new VM.

If you intend to use your backed up data only to restore existing VDIs and VMs, you can skip this step.

3. Get a reference for the VDI you want to snapshot:

```
vdi_ref = session.xenapi.VDI.get_by_uuid("<vdi_uuid>")
```

4. Enable changed block tracking for the VDI:

```
session.xenapi.VDI.enable_cbt(<vdi_ref>)
```

For more information, see Using changed block tracking with a virtual disk image.

5. Take a snapshot of the VDI:

```
base_snapshot_vdi_ref = session.xenapi.VDI.snapshot(<vdi_ref>)
```

This VDI snapshot is the base snapshot.

6. Export the base VDI snapshot to the backup location. This can be done by using HTTPS or by using the command line.

For example, at the xe command line run:

```
xe vdi-export uuid=<base-snapshot-vdi-uuid> filename=<name of
    export>
```

Or, in Python, you can use the following code:

```
session_id = session._session
url = ('https://%s/export_raw_vdi?session_id=%s&vdi=%s&format=raw'
        % (<xs_host>, session_id, <base_snapshot_vdi_uuid>))
with requests.Session() as http_session:
    urllib3.disable_warnings(urllib3.exceptions.
        InsecureRequestWarning)
    request = http_session.get(url, verify=False, stream=True)
    with open(<export_path>, 'wb') as filehandle:
        shutil.copyfileobj(request.raw, filehandle)
    request.raise_for_status()
```

Where <*export_path*> is the location you want to write the exported VDI to.

7. Optional: For each VDI snapshot, delete the snapshot data, but retain the metadata:

```
session.xenapi.VDI.data_destroy(<base_snapshot_vdi_ref>)
```

This frees up space on the host or SR.

For more information, see Deleting VDI snapshot data and retaining the snapshot metadata.

Taking incremental backups

After taking the initial VDI snapshot and exporting all the data, the following steps can be repeated every time an incremental backup is taken of the VDI.
These incremental backups export only the blocks that have changed since the previous snapshot was taken.

To take an incremental backup, complete the following steps:

1. Check that changed block tracking is enabled:

```
is_cbt_enabled = session.xenapi.VDI.get_cbt_enabled(<vdi_ref>)
```

If the value of `is_cbt_enabled` is not **`true`**, you must complete the steps in the *Setting up changed block tracking* section, before taking incremental backups.
For more information, see Incremental backup sets.

If changed block tracking is disabled and this is unexpected, this state might indicate that the host or SR has crashed since you last took an incremental backup or that a XenServer user has disabled changed block tracking.

2. Take a snapshot of the VDI:

```
snapshot_vdi_ref = session.xenapi.VDI.snapshot(<vdi_ref>)
```

3. Compare this snapshot to a previous snapshot to find the changed blocks:

```
bitmap = session.xenapi.VDI.list_changed_blocks(<
    base_snapshot_vdi_ref>, <snapshot_vdi_ref>)
```

This call returns a base64-encoded bitmap that indicates which blocks have changed.
For more information, see Get the list of blocks that changed between VDIs.

4. Get details for a list of connections that can be used to use to access the VDI snapshot over the NBD protocol.

   ```
   connections = session.xenapi.VDI.get_nbd_info(<snapshot_vdi_ref>)
   ```

   This call returns a list of connection details that are specific to this session.
   Each set of connection details in the list contains a dictionary of the parameters required for an NBD client connection.
   For more information, see Getting NBD connection information for a VDI.

   > **Note:**
   >
   > Ensure that this session with the host remains logged in until after you have finished reading from the network block device.

5. From your NBD client, complete the following steps to export the changed blocks to the backup location.
   For example, when using the Linux `nbd-client`:

   a) Connect to the NBD server.

   ```
   nbd-client <address> <port> -N <exportname> -cacertfile <
       cacert> -tlshostname <subject>
   ```

   - The <*address*>, <*port*>, <*exportname*>, and <*subject*> values passed as parameters to the connection command are the values returned by the `get_nbd_info` call.
   - The <*cacert*> is a file containing one or more trusted Certificate Authority certificates of which at least one has signed the NBD server's TLS certificate.
     That TLS certificate is included in the values returned by the `get_nbd_info` call.
     If the TLS certificate returned by the `get_nbd_info` call is self-signed, it can be used as the value of `cacert` here to authenticate itself.

     For more information about using these values, see Getting NBD connection information for a VDI.

   b) Read off the blocks that are marked as changed in the bitmap returned from step 3.

   c) Disconnect from the block device:

   ```
   nbd-client -d <block_device>
   ```

   d) Optional: We recommend that you retain the bitmap associated with each changed block export at your backup location.

To complete the preceding steps, you can use any NBD client implementation that supports the "fixed newstyle"version of the NBD protocol.
For more information, see Exporting the changed blocks using an NBD clientl)l).

6. Optional: On the host, delete the VDI snapshot, but retain the metadata:

```
session.xenapi.vdi.data_destroy(<snapshot_vdi_ref>)
```

   This frees up space on the host or SR.

   For more information, see Deleting VDI snapshot data and retaining the snapshot metadata.

Restoring a VDI from exported changed block data

When you want to use your incremental backups to restore or import data from a VDI, you cannot use individual exports of changed blocks to do this.
You must first coalesce the exported changed blocks onto a base snapshot.
Use this coalesced VDI to restore or import backed up data.

1. Create a coalesced VDI.

   For each set of exported changed blocks between the base snapshot and the snapshot you want to restore to, create a coalesced VDI from a previous base VDI and the subsequent set of exported changed blocks.
   Ensure that you apply sets of the changed blocks to the base VDI in the order that they were snapshotted.

   To create a coalesced VDI from a base VDI and the subsequent set of exported changed blocks, complete the following steps:

   a) Get the bitmap that was used in step 3 to derive this set of exported changed blocks.

   b) For each block in the VDI:

      - If the bitmap indicates that the block has changed, read the block data from the set of exported changed blocks and append that data to the coalesced VDI.

      - If the bitmap indicates that the block has not changed, read the block data from the base VDI and append that data to the coalesced VDI.

   c) Use the coalesced VDI as the base VDI for the next iteration of this step.
      Or, if you have reached the target snapshot level, use this coalesced VDI in the next step to restore a VDI in XenServer.

   For more information, see Coalescing changed blocks onto a base VDI.

   You can now use this coalesced VDI to either import backed up data into a new VDI or to restore an existing VDI.

2. Optional: Create a new VM and new VDI.

   You can create a new VM and new VDI to import the coalesced VDI into.
   However, if you intend to use the coalesced VDI to restore an existing VDI, you can skip this step.

   To create a new VM and new VDI, complete the following steps

   a) Create a new VDI:

   ```
   vdi_record = {

        "SR": <sr>,
        "virtual_size": <size>,
        "type": "user",
        "sharable": False,
        "read_only": False,
        "other_config": {
    }
   ,
        "name_label": "<name_label>"
   }

   vdi_ref = session.xenapi.VDI.create(vdi_record)
   vdi_uuid = session.xenapi.VDI.get_uuid(vdi_ref)
   ```

   Where <*sr*> is a reference to the SR that the original VDI was located on and <*size*> is the size of the original VDI.

   b) To create a new VM that uses the VDI created in the previous step, import the VM metadata associated with the snapshot level you are using to restore the VDI:

   ```
   vdi_string = "&vdi:%s=%s" % (<original_vdi_uuid>, <
       new_vdi_uuid>)

   task_ref = session.xenapi.task.create("import_vm", "Task to
       track vm import")

   url = ('https://%s/import_metadata?session_id=%s&task_id=%s%s'
          % (host, session._session, task_ref, vdi_string))

   with open(<vm_import_path>, 'r') as filehandle:
       urllib3.disable_warnings(urllib3.exceptions.
           InsecureRequestWarning)
       with requests.Session() as http_session:
           request = http_session.put(url, filehandle, verify=
               False)
           request.raise_for_status()
   ```

   Where <*vm_import_path*> is the location of the VM metadata.

   The `vdi:` query parameter changes the VM from pointing to its original VDI to pointing to the new VDI created in the previous step.

You might want to create multiple new VDIs.
If you want to change multiple VDI references for your new VM, add a `vdi:` query parameter for each VDI to the import URL.

The new VM is created from the imported metadata and its VDI reference is updated to point at the VDI created in the previous step.
You can extract a reference to this new VM from the result of the task.
For more information, see the samples on GitHub.

3. Import the coalesced VDI snapshot to the XenServer host at the UUID of the VDI you want to replace with the restored version.
   This VDI can be either an existing VDI or the VDI created in the previous step.

   In Python, you can use the following code:

```
session_id = session._session
url = ('https://%s/import_raw_vdi?session_id=%s&vdi=%s&format=%s'
       % (<xs_host>, session_id, <vdi_uuid>, 'raw'))
with open(<import_path>, 'r') as filehandle:
    urllib3.disable_warnings(urllib3.exceptions.
        InsecureRequestWarning)
    with requests.Session() as http_session:
        request = http_session.put(url, filehandle, verify=False)
        request.raise_for_status()
```

   Where <*vdi_uuid*> is the UUID of the VDI you want to overwrite with the restored data from the coalesced VDI and <*import_path*> is the location of the coalesced VDI.

Enabling NBD connections on XenServer

May 10, 2023

XenServer acts as a network block device (NBD) server and makes VDI snapshots available over NBD connections.
However, to connect to XenServer over an NBD connection, you must enable NBD connections for one or more networks.

> **Important**
>
> We recommend that you use a dedicated network for your NBD traffic.

By default, NBD connections are not enabled on any networks.

> **Note:**
>
> Networks associated with a XenServer pool that have NBD connections enabled must either all have the `nbd` purpose or all have the `insecure_nbd` purpose. You cannot have a mix of normal NBD networks (`FORCEDTLS`) and insecure NBD networks (`NOTLS`).
> To switch the purpose of all networks, you must first disable normal NBD connections on all networks before enabling either normal or insecure NBD connections on any networks.

Enabling an NBD connection for a network (FORCEDTLS mode)

We recommend that you use TLS in your NBD connections.
When NBD connections with TLS are enabled, any NBD clients that attempt to connect to XenServer must use TLSv1.2.
The NBD server runs in `FORCEDTLS` mode with the "fixed newstyle"NBD handshake.
For more information, see the NBD protocol documentation.

To enable NBD connections with TLS, use the `purpose` parameter of the network.
Set this parameter to include the value `nbd`.
Ensure that you wait for the setting to propagate before attempting to use this network for NBD connections.
The time it takes for the setting to propagate depends on your network and is at least 10 seconds.
We recommend that you use a retry loop when making the NBD connection.

Examples

You can use any of our supported language bindings to enable NBD connections.
The following examples show how to do it in Python and at the xe command line.

Python:

```
session.xenapi.network.add_purpose(<network_ref>, "nbd")
<!--NeedCopy-->
```

xe command line:

```
xe network-param-add param-name=purpose param-key=nbd uuid=<network-
    uuid>
<!--NeedCopy-->
```

Enabling an insecure NBD connection for a network (NOTLS mode)

We recommend that you do not enable insecure NBD connections.
Instead use `FORCEDTLS` NBD connections.

However, the ability to connect to the XenServer over an insecure NBD connection is provided for development and testing with the NBD server operating in `NOTLS` mode as described in the NBD protocol.

To enable insecure NBD connections, use the `purpose` parameter of the network.
Set this parameter to include the value `insecure_nbd`.
Ensure that you wait for the setting to propagate before attempting to use this network for NBD connections.
The time it takes for the setting to propagate depends on your network and is at least 10 seconds.
We recommend that you use a retry loop when making the NBD connection.

Examples

You can use any of our supported language bindings to enable insecure NBD connections.
The following examples show how to do it in Python and at the xe command line.

Python:

```
session.xenapi.network.add_purpose(<network_ref>, "insecure_nbd")
<!--NeedCopy-->
```

xe command line:

```
xe network-param-add param-name=purpose param-key=insecure_nbd uuid=<
    network-uuid>
<!--NeedCopy-->
```

Disabling NBD connections for a network

To disable NBD connections for a network, remove the NBD values from the `purpose` parameter of the network.

Examples

You can use any of our supported language bindings to disable NBD connections.
The following examples show how to do it in Python and at the xe command line.

Python:

```
session.xenapi.network.remove_purpose(<network_ref>, "nbd")
<!--NeedCopy-->
```

Or, for insecure NBD connections:

```
session.xenapi.network.remove_purpose(<network_ref>, "insecure_nbd")
<!--NeedCopy-->
```

xe command line:

```
xe network-param-remove param-name=purpose param-key=nbd uuid=<network-
    uuid>
<!--NeedCopy-->
```

Or, for insecure NBD connections:

```
xe network-param-remove param-name=purpose param-key=insecure_nbd uuid
    =<network-uuid>
<!--NeedCopy-->
```

Using changed block tracking with a virtual disk image

May 10, 2023

The changed block tracking capability can be enabled and disabled for individual virtual disk images (VDIs).

Incremental backup sets

When you enable changed block tracking for a VDI you start a new set of incremental backups for that VDI.
The first action you must take when starting a set of incremental backups is to create a baseline snapshot and to backup its full data.

After you disable changed block tracking, or after changed block tracking is disabled by XenServer or a user, no further incremental backups can be added to this set.
If changed block tracking is enabled again, you must take another baseline snapshot and start a new set of incremental backups.

You cannot compare VDI snapshots taken as part of one set of incremental backups with VDI snapshots taken as part of a different set of incremental backups.
If you attempt to list the changed blocks between snapshots that are part of different sets, you get an error with the message `Source and target VDI are unrelated`.

You can use some or all of the data in previous incremental backup sets to create VDIs that you can use to restore the state of a VDI.
For more information, see Coalescing changed blocks onto a base VDI.

Enabling changed block tracking for a VDI

By default, changed block tracking is not enabled for a VDI.
You can enable changed block tracking by using the `enable_cbt` call.

When changed block tracking is enabled for a VDI, additional log files are created on the SR to list the changes since the last backup.
Blocks of 64 kB within the VDI are tracked and changes to these blocks recorded in the log layer.

The associated VM remains in the same state as before changed block tracking was enabled.

To enable changed block tracking, an SR must be attached, be writable, and have enough free space for the log files to be created on it.
The associated VM can be in any state when changed block tracking is enabled or disabled.
It is not required that the VM be offline.

Changed block tracking can only be enabled for a VDI that is one of the following types:

- user
- system

In addition, if the `VDI.on_boot` field is set to `reset`, you cannot enable changed block tracking for the VDI.

Examples

You can use any of our supported language bindings to enable changed block tracking for a VDI.
The following examples show how to do it in Python and at the xe command line.

Python:

```
session.xenapi.VDI.enable_cbt(<vdi_ref>)
<!--NeedCopy-->
```

xe command line:

```
xe vdi-enable-cbt uuid=<vdi-uuid>
<!--NeedCopy-->
```

Errors

You might see the following errors when using this call:

VDI_MISSING:

- The call cannot find the VDI.

  Check that the reference or UUID you are using to refer to the VDI is correct. Check that the VDI exists.

VDI_INCOMPATIBLE_TYPE:

- The VDI is of a type that does not support changed block tracking.

  Check that the type of the VDI is `system` or `user`.
  You can use the `get_type` call to find out the type of a VDI.
  If your VDI is an incompatible type, you cannot enable changed block tracking.
  For more information, see Checking the type of a VDI or VDI snapshot.

VDI_ON_BOOT_MODE_INCOMPATIBLE_WITH_OPERATION:

- The value of the `on_boot` field of the VDI is set to `reset`.

  Check the value of the `on_boot` field by using the `get_on_boot` call.
  If appropriate, you can use the `set_on_boot` call to change the value of this field to `persist`.

SR_NOT_ATTACHED, SR_HAS_NO_PBDS:

- The call cannot find an attached SR.

  Check that there is an SR attached to the host and that the SR is writable.
  You cannot enable changed block tracking unless the host has access to an SR to which the changed block information can be written.

If you attempt to enable changed block tracking for a VDI that already has changed block tracking enabled, no error is thrown.

Disabling changed block tracking for a VDI

You can disable changed block tracking for a VDI by using the `disable_cbt` call.

When changed block tracking is disabled for a VDI, the active disks are detached and reattached without the log layer.
The associated VM remains in the same state as before changed block tracking was disabled.

Examples

You can use any of our supported language bindings to disable changed block tracking for a VDI.
The following examples show how to do it in Python and at the xe command line.

Python:

```
session.xenapi.VDI.disable_cbt(<vdi_ref>)
<!--NeedCopy-->
```

xe command line:

```
xe vdi-disable-cbt uuid=<vdi-uuid>
<!--NeedCopy-->
```

Errors

You might see the same sorts of errors for this call and you might for the `enable_cbt` call.

If you attempt to disable changed block tracking for a VDI that already has changed block tracking disabled, no error is thrown.

Checking whether changed block tracking is enabled

The value of the boolean `cbt_enabled` VDI field shows whether changed block tracking is enabled for that VDI.
You can query the value of this field by using the `get_cbt_enabled` call.

A return value of **`true`** indicated that changed block tracking is enabled for this VDI.

Examples

You can use any of our supported languages to check whether a VDI has changed block tracking enabled.
The following examples show how to do it in Python and at the xe command line.

Python:

```
is_cbt_enabled = session.xenapi.VDI.get_cbt_enabled(<vdi_ref>)
<!--NeedCopy-->
```

xe command line:

```
xe vdi-param-get param-name=cbt-enabled uuid=<vdi-uuid>
<!--NeedCopy-->
```

Deleting VDI snapshot data and retaining the snapshot metadata

May 10, 2023

A VDI snapshot is made up of both data and metadata.
The data is the full image of the disk at the time the snapshot was taken.
The metadata includes the changed block tracking information.

After the snapshot data on the host has been exported to the backup location, you can use the `data_destroy` call to delete only the snapshot data and retain only the snapshot metadata on the host.
This action converts the snapshot that is stored on the host or SR into a smaller metadata-only snapshot.
The `type` field of the snapshot changes to be `cbt_metadata`.

Metadata-only snapshots are linked to the metadata-only snapshots that precede and follow them in time.

You can use the `data_destroy` call only for snapshots for VDIs that have changed block tracking enabled.

> **Note:**
>
> The API also provides a `destroy` call, which deletes both the data in the snapshot and the metadata in the snapshot.

Do not use the `destroy` call to delete snapshots that are part of a set of changed block tracking backups unless you are sure that you no longer need the changed block tracking metadata.

For example, use `destroy` to remove a metadata-only snapshot that is older than age allowed by your retention policy.

Examples

You can use any of our supported language bindings to delete the data in a snapshot and convert the snapshot to a metadata only snapshot.
The following examples show how to do it in Python and at the xe command line.

Python:

```
session.xenapi.VDI.data_destroy(<snapshot_vdi_ref>)
<!--NeedCopy-->
```

xe command line:

```
xe vdi-data-destroy uuid=<snapshot_vdi_uuid>
<!--NeedCopy-->
```

Errors

You might see the following errors when using this call:

VDI_MISSING:

- The call cannot find the VDI snapshot.

  Check that the reference or UUID you are using to refer to the VDI snapshot is correct. Check that the VDI exists.

VDI_NO_CBT_METADATA:

- No changed block tracking metadata exists for this VDI snapshot.

  Check that changed block tracking is enabled for the VDI.
  You cannot use the `data_destroy` call on VDIs that do not have changed block tracking enabled.
  For more insformation, see Using changed block tracking with a virtual disk image.

VDI_IN_USE:

- The VDI snapshot is currently in use by another operation.

  Check that the VDI snapshot is not being accessed by another client or operation.
  Check that the VDI is not attached to a VM.

  If the VDI snapshot is connected to a VM snapshot by a VBD, you receive this error.
  Before you can run `VDI.data_destroy` on this VDI snapshot, you must remove the VM snapshot.
  Use `VM.destroy` to remove the VM snapshot.

Checking the type of a VDI or VDI snapshot

The value of the `type` VDI field shows what type of VDI or VDI snapshot an object is.
The values this field can have are stored in the `vdi_type` enum.
You can query the value of this field by using the `get_type` call.

A metadata-only VDI snapshot has the type `cbt_metadata`.

Examples

You can use any of our supported language bindings to query the VDI type of a VDI or VDI snapshot.
The following examples show how to do it in Python and at the xe command line.

Python:

```
vdi_type = session.xenapi.VDI.get_type(<snapshot_vdi_ref>)
<!--NeedCopy-->
```

xe command line:

```
xe vdi-param-get param-name=type uuid=<snapshot_vdi_uuid>
<!--NeedCopy-->
```

Getting the list of blocks that changed between VDIs

May 10, 2023

You can use the `list_changed_blocks` call to get a list of the blocks that have changed between two VDIs.
Both VDI snapshots must be taken after changed block tracking is enabled on the VDI.

This call takes as parameters references to two VDI snapshots:

- `VDI_from`: The earlier VDI snapshot.
- `VDI_to`: The later VDI snapshot. This VDI *cannot* be attached to a VM at the time this comparison is made.

This operation does not require the VM associated with the VDIs to be offline at the time the comparison is made.

The changed blocks are listed in a base64-encoded bitmap.
Each bit in the bitmap indicates whether a 64 kB block in the VDI has been changed in comparison to an earlier snapshot.
A bit set to 0 indicates that the block is the same.
A bit set to 1 indicates that the block has changed.

The bit in the first position in the bitmap represents the first block in the VDI.
For example, if the bitmap is 01100000, this indicates that the first block has not changed, the second and third blocks have changed, and all other blocks have not changed.

Examples

You can use any of our supported languages to get the bitmap that lists the changed blocks between two VDI snapshots.
The following examples show how to do it in Python and at the xe command line.

Python:

```
bitmap = session.xenapi.VDI.list_changed_blocks(<
    previous_snapshot_vdi_ref>, <new_snapshot_vdi_ref>)
<!--NeedCopy-->
```

You can convert the base64-encoded bitmap this call returns into a human-readable string of 1s and 0s:

```
from bitstring import BitStream
import base64
data = BitStream(bytes=base64.b64decode(bitmap))
<!--NeedCopy-->
```

xe command line:

```
xe vdi-list-changed-blocks vdi-from-uuid=<previous_snapshot_vdi_uuid>
    vdi-to-uuid=<new_snapshot_vdi_uuid>
<!--NeedCopy-->
```

Errors

You might see the following errors when using this call:

VDI_MISSING:

- The call cannot find one or both of the VDI snapshots.

  Check that the reference or UUID you are using to refer to the VDI snapshot is correct.
  Check that the VDI snapshot exists.

VDI_IN_USE:

- The VDI snapshot is currently in use by another operation.

  Check that the VDI snapshot is not being accessed by another client or operation.
  Check that the more recent VDI snapshot is not attached to a VM.
  The newer VDI in the comparison cannot be attached to a VM at the time of the comparison.

Source and target VDI are unrelated:

- The VDI snapshots are not linked by changed block metadata.

  You can only list changed blocks between snapshots that are taken as part of the same set of incremental backups.
  For more information, see Incremental backup sets.

Export changed blocks over a network block device connection

May 10, 2023

XenServer runs an NBD server on the host that can make VDI snapshots accessible as a network block device to NBD clients.
The NBD server listens on port 10809 and uses the "fixed newstyle"NBD protocol.
For more information, see the NBD protocol documentation.

NBD connections must be enabled for one or more of the XenServer networks before you can export the changed blocks over NBD.
For more information, see Enabling NBD connections on XenServer.

Getting NBD connection information for a VDI

From a logged in XenAPI session, you can use the `get_nbd_info` call to get a list of connection details for a VDI snapshot made available as a network block device.

These connection details are specific to the session that creates them and the NBD client uses this logged in session when making its connection.
Any set of connection details in the list can be used by the NBD client when accessing the VDI snapshot.

Each set of connection details in the list is provided as a dictionary containing the following information:

`address`:

- The IP address (IPv4 or IPv6) of the NBD server.

`port`:

- The TCP port to connect to the XenServer NBD server on.

`cert`:

- The TLS certificate used by the NBD server encoded as a string in PEM format.
  When XenServer is configured to enable NBD connections in `FORCEDTLS` mode, the server presents this certificate during the TLS handshake and the NBD client must verify the server TLS certificate against this TLS certificate.
  For more information, see "Verifying TLS certificates for NBD connections".

`exportname`:

- A token that the NBD client can use to request the export of a VDI from the NBD server.
  The NBD client provides the value of this token to the NBD server using the `NBD_OPT_EXPORT_NAME` option during the NBD option haggling phase of an NBD connection.

  This token contains a reference to a logged in XenAPI session.
  The XenAPI session must remain logged in for this token to continue to be valid.
  Because the token contains a reference to a XenAPI session, you must handle the token securely to prevent the session being hijacked.

  The format of this token is not guaranteed and might change in future releases of XenServer.
  Treat the export name as an opaque token.

`subject`:

- A subject of the TLS certificate returned as the value of `cert`. This field is provided as a convenience.

Examples

You can use any of our supported languages to get the list of NBD connection details for a VDI snapshot. The following examples show how to do it in Python.

Python:

```
connection_list = session.xenapi.VDI.get_nbd_info(<snapshot_vdi_ref>)
<!--NeedCopy-->
```

This call requires a logged in XenAPI session that remains logged in while the VDI snapshot is accessed over NBD.
This means that this command is not available at the xe command line.

Errors

You might see the following errors when using this call:

VDI_INCOMPATIBLE_TYPE:

- The VDI is of a type that does not support being accessed as a network block device.

  Check that the type of the VDI is not `cbt_metadata`.
  You can use the `get_type` call to find out the type of a VDI.
  If your VDI is `cbt_metadata`, you cannot access it as a network block device. For more information, see Checking the type of a VDI or VDI snapshot.

An empty list of connection details:

- The VDI cannot be accessed.

  Check that the XenServer host that runs the NBD server has a PIF with an IP address.

  Check that you have at least one network in your pool with the purpose `nbd` or `insecure_nbd`
  .

  For more information, see Enabling NBD connections on XenServer.

  Check that storage repository the VDI is on is attached to a host that is connected to one of the NBD-enabled networks.

Exporting the changed blocks using an NBD client

An NBD client running in the backup location can connect to the NBD server that runs on the XenServer host and access the VDI snapshot by using the provided connection details.

The NBD client that you use to connect to the XenServer NBD server can be any implementation that supports the "fixed newstyle"version of the NBD protocol.

When choosing or developing an NBD client implementation, consider the following requirements:

- The NBD client must support the "fixed newstyle"version of the NBD protocol.
  For more information, see the NBD protocol documentation.
- The NBD client must request an export name returned by the `get_nbd_info` call that corresponds to an existing logged in XenAPI session.
  The client makes this request by using the `NBD_OPT_EXPORT_NAME` option during the NBD option haggling phase of the NBD connection.
- The NBD client must verify the TLS certificate presented by the NBD server by using the information returned by the `get_nbd_info` call.
  For more information, see "Verifying TLS certificates for NBD connections".

> **Note:**
>
> If you are using the Linux upstream NBD client, a minimum version of 3.15 is required to support TLS.

After the NBD client has made a connection to the XenServer host and accessed the VDI snapshot, you can use the bitmap provided by the `list_changed_blocks` call to select which blocks to read. For more information, see Getting the list of blocks that changed between VDIs.

> **Note:**
>
> XenServer supports up to 16 concurrent NBD connections.

Verifying TLS certificates for NBD connections

When connecting to the NBD server using TLS, the NBD client must verify the certificate that the server presents as part of the TLS handshake.

We recommend that you use one of the following methods of verification depending on your NBD client implementation:

- Verify that the server certificate matches the certificate returned by the `get_nbd_info` call.
- Verify that the public key of the server certificate matches the public key of the certificate returned by the `get_nbd_info` call.

Alternative approach As a less preferred option, it is possible for the NBD client to verify the certificate that the server presents during the TLS handshake by checking that the certificate meets all of the following criteria:

- It is signed by a trusted Certificate Authority
- It has an `Alternative Subject Name` (or, if absent, a `Subject`) that matches the subject returned by the `get_nbd_info` call.

Coalescing changed blocks onto a base VDI

May 10, 2023

When using backed up data to restore the state of a VDI, you must import a full VDI into XenServer. You cannot import only the sets of changed blocks.
To import incremental backups created with changed block tracking data into XenServer, you must first use these incremental backups to create a full VDI.

A set of incremental backups created with changed block tracking can be used to create a full VDI whose data is identical to the source VDI at the time an incremental backup was taken.

For more information about incremental backup sets, see Incremental backup sets.

For example, you have a set of incremental backups that comprises:

- A base snapshot that captures the data for the full VDI.
- Backup 1: The first incremental backup, which consists of a bitmap list of blocks changed since the base snapshot and the data for only those changed blocks.
- Backup 2: The second incremental backup, which consists of a bitmap list of blocks changed since backup 1 and the data for only those changed blocks.

If you want to restore a VDI on XenServer to the state it was at when backup 2 was taken, you must create a VDI that takes blocks from the base snapshot, changed blocks from backup 1, and changed blocks from backup 2.
To do this, you can apply each set of changed blocks in sequence to the base snapshot of the VDI.

First build up a coalesced VDI by taking unchanged blocks from the base snapshot and changed blocks from those exported at backup 1.
The bitmap list of changed blocks that was used to create backup 1 defines which blocks are changed.

After coalescing the base snapshot with the changed blocks exported at backup 1, you have a full VDI whose data is identical to that of the source VDI at the time backup 1 was taken. Call this coalesced VDI "VDI 1".

Next, use this coalesced VDI, VDI 1, to create another coalesced VDI by taking unchanged blocks from VDI 1 and changed blocks from those exported at backup 2.
The bitmap list of changed blocks that was used to create backup 2 defines which blocks are changed.

After coalescing VDI 1 with the changed blocks exported at backup 2, you have a full VDI whose data is identical to that of the source VDI at the time backup 2 was taken.
Call this coalesced VDI "VDI 2".

This coalesced VDI, VDI 2, can be used to restore the state of the VDI on XenServer at the time that a snapshot was taken for backup 2.

When creating a coalesced VDI, ensure that you work with your VDIs and changed blocks as binary.

Ensure that you verify the integrity of the backed up and restored VDIs.
For example, you can do this by computing the checksums of the data.

Examples

The following example shows how to create a coalesced VDI.
The example shows applying a single set of changed blocks to the base VDI snapshot.
To apply multiple sets of changed blocks, you must repeat this process for each set of changed blocks in order from oldest to most recent, using the output from the previous iteration as the base VDI for the next iteration.

Python:

```python
def write_changed_blocks_to_base_VDI(vdi_path, changed_block_path,
    bitmap_path, output_path):
    bitmap = open(bitmap_path, 'r')
    vdi = open(vdi_path, 'r+b')
    blocks = open(changed_block_path, 'r+b')
    combined_vdi = open(output_path, 'wb')

    try:
        bitmap_r = bitmap.read()
        cb_offset = 0
        for x in range(0, len(bitmap_r)):
            offset = x * changed_block_size
            if bitmap_r[x] == "1":
                blocks.seek(cb_offset)
                blocks_r = blocks.read(changed_block_size)
                combined_vdi.write(blocks_r)
                cb_offset += changed_block_size
            else:
                vdi.seek(offset)
                vdi_r = vdi.read(changed_block_size)
                combined_vdi.write(vdi_r)
<!--NeedCopy-->
```

Troubleshoot Changed Block Tracking

May 10, 2023

This article includes some common error scenarios you might encounter while enabling and using changed block tracking.

Common error scenarios

You can't enable changed block tracking on a VDI

- Ensure that the VDI is not a snapshot or a raw VDI. You can't enable changed block tracking on snapshots or raw VDIs.

You can't data destroy a VDI

- Ensure that the VDI you are referring to is a snapshot.
- Ensure that changed block tracking is enabled. Otherwise, you might see the `VDI_NO_CBT_METADATA` error.
  - In XenCenter, check the **Storage** tab to see if changed block tracking is enabled for the VDI you took a snapshot of.
  - Or, use the xe CLI to check the `cbt_enabled` field of the VDI snapshot: `xe vdi-param-list uuid=<snapshot_uuid>`

- Check whether a `<snapshot_uuid>.cbtlog` file exists in the storage repository:
    - For LVM-based storage repositories, run the command `lvs` to check whether a `<snapshot_uuid>.cbtlog` file exists in the storage repository.
    - For file-based storage repositories, look for the files in the location `/run/sr-mount/<sr-uuid>/<snapshot-uuid>.cbtlog`.

You can't list changed blocks between two VDI snapshots

- Ensure that the VDIs are snapshots. If the `vdi_to` VDI is not a snapshot, ensure it is not attached.
- Ensure that both VDI snapshots have changed block tracking enabled.
- Ensure that the VDI snapshots are related and in the right order.

  You can use the `cbt-util` utility, which helps establish chain relationship. If the VDI snapshots are not linked by changed block metadata, you get errors like "SR_BACKEND_FAILURE_460", "Failed to calculate changed blocks for given VDIs", and "Source and target VDI are unrelated".

  Example usage of cbt-util:

  ```
  cbt-util get –c –n <name of cbt log file>
  ```

  The `–c` option prints the child log file UUID.

Notable error conditions

Changed block tracking is disabled when certain errors are encountered, for example:

- The changed block tracking log is found to be inconsistent at the time of attaching a VDI. This situation can occur when a XenServer host or SR crashes.
- The resize of a changed block tracking log file was unsuccessful on the source VDI resize.
- Insufficient space remains on disk to create a changed block tracking log file when changed block tracking is activated or a snapshot is created.

However, the primary action (for example, attach, resize, or snapshot) does succeed in those error conditions and a log message is logged in SMlog. Also, an alert is generated in XenCenter to notify you that changed block tracking is disabled.

Troubleshooting the NBD server

XenServer acts as network block device (NBD) server and makes VDI snapshots available over NBD connections. For more information, see Enabling NBD connections on XenServer.

Notes:

- The NBD server logs are in `/var/log/daemon.log`.
- NBD connections work with all network configurations, including VLAN, bond network, and VLAN on bond.
- NBD connections don't work when an NBD client is in dom0 on the same host as the NBD server.
- Networks associated with a XenServer pool that have NBD connections enabled must either all have the purpose `nbd` or all have the purpose `insecure_nbd`. You cannot have a mix of normal NBD networks (FORCEDTLS) and insecure NBD networks (NOTLS).
- The service (xapi-nbd) shows the state "failed"after it stopped. This state does not indicate an error.

Common NBD connection issues

The command `get_nbd_info` returns an empty list of connection details

- Ensure that the XenServer host that runs the NBD server has a PIF with an IP address.
- Ensure that you have at least one network in your pool with the purpose `nbd` or `insecure_nbd`.
- Ensure that the storage repository that the VDI is on is attached to a host that is connected to one of the NBD-enabled networks.

You can't access the IP address returned by the command `get_nbd_info`

- Check whether NBD is enabled on a network that is not reachable by the client.
- Check whether multiple networks are mixed on the same subnet and NBD is allowed on some of them but blocked on others.
- Check the NBD network configuration.

The connection refused or closed, or the NBD client hangs

- Verify whether the session that the client passes to the NBD server is valid. If the session has expired or is not valid, you see the error "SESSION_INVALID". The session has to be valid for the time of the backup, otherwise, the NBD server refuses the connection or the client might hang.
- The maximum parallel connection limit might have been exceeded. To work around this, you can restart xapi-nbd.
- If the NBD server is configured to use TLS, ensure that the client is able to use TLS.

Appendices

July 3, 2023

Constraints

The following section lists advisories and constraints to consider when using changed block tracking.

- Changed block tracking is available only to customers with a Premium Edition license for XenServer.
  If a customer without a Premium Edition license attempts to use an incremental backup solution for XenServer that uses changed block tracking, they are prevented from enabling changed block tracking on new VDIs.
  However, if the customer has existing VDIs with changed block tracking enabled, they can still perform other changed block tracking actions on these VDIs.
- Changed block tracking information is lost on storage live migration.
  If you attempt to migrate a VM that has VDIs with changed block tracking enabled, you are prevented from doing so.
  You must disable changed block tracking before storage live migration is allowed.
- If a host or an SR crashes, XenServer disables changed block tracking for all VDIs on that host or SR.
  Before taking a VDI snapshot, we recommend that you check whether changed block tracking is enabled.
  If changed block tracking is disabled and this is not expected, this can indicate that a crash has occurred or that a XenServer user has disabled changed block tracking.

  To continue using changed block tracking, you must enable changed block tracking again and create a new baseline by taking a full VDI snapshot.
  Subsequent changed block tracking metadata uses this snapshot as a new baseline.

  The set of snapshots and changed block tracking data captured before the crash cannot be used as a baseline or comparison for any snapshots taken after the crash.
  However, the set of incremental backups taken before the crash can be used to create a VDI image to use to restore the VDI to a previous state.

  For more information, see Incremental backup sets.
- XenServer supports a maximum of 16 concurrent NBD connections.
- Changed block tracking is not supported for VDIs stored on thin-provisioned shared GFS2 block storage.

Additional Resources

The following resources provide additional information:

- GitHub repository of sample code
- XenServer Management API Guide
- XenServer Software Development Kit Guide
- NBD protocol documentation
- nbd-client manpage

XenServer Supplemental Packs and the DDK Guide

May 10, 2023

Supplemental packs are used to modify and extend the functionality of a XenServer host by installing software into the control domain, dom0.
For example, an OEM partner might want to ship XenServer with a suite of management tools that require SNMP agents to be installed, or provide a driver that supports the latest hardware.
Users can add supplemental packs either during initial XenServer installation, or at any time afterwards.
Facilities also exist for OEM partners to add their supplemental packs to the XenServer installation repositories, in order to allow automated factory installations.

Purpose of supplemental packs

Supplemental packs consist of a number of packages along with information describing their relationship to other packs.
Individual packages are in the Red Hat RPM file format, and must be able to install and uninstall cleanly on a fresh installation of XenServer.

Packs are created using the XenServer Driver Development Kit (DDK).
This has been extended to not only allow the creation of supplemental packs containing only drivers (also known as driver disks), but also packs containing userspace software to be installed into dom0.

Examples and tools are included in the XenServer DDK to help developers create their own supplemental packs.
However, for partners who want to integrate pack creation into their existing build environments, only a few scripts taken from the DDK are necessary.

Why a separate DDK?

XenServer is based on a standard Linux distribution, but for performance, maintainability, and compatibility reasons ad-hoc modifications to the core Linux components are not supported.
As a result, operations that require recompiling drivers for the Linux kernel require formal guidance from Citrix, which the DDK provides.
In addition, the DDK provides the necessary compile infrastructure to achieve this, whereas a XenServer installation does not.

XenServer integrates the latest device support from kernel.org on a regular basis to provide a current set of device drivers.
However, assuming appropriate redistribution agreements can be reached, there are situations where including additional device drivers in the shipping XenServer product, such as drivers not available through kernel.org, or drivers that have functionality not available through kernel.org, is greatly beneficial to joint customers of Citrix and the device manufacturer.
The same benefits can apply by supplying device drivers independent of the XenServer product.

In addition, components such as command line interfaces (CLIs) for configuring and managing devices are also very valuable to include in the shipped XenServer product.
Some of these components are simple binary RPM installs, but in many cases they are combined with the full driver installation making them difficult or impossible for
administrators to install into XenServer.
In either case including current versions of everything the administrator requires to use the device on XenServer in a supplemental pack provides significant value.

The DDK allows driver vendors to perform the necessary packaging and compilation steps with the XenServer kernel, which is not possible with the XenServer product alone.
Supplemental packs can be used to package up both drivers and userspace tools into one convenient ISO that can be easily installed by XenServer users.

Benefits

Supplemental packs have a variety of benefits over and above partners producing their own methods for installing add-on software into XenServer:

- Integration with the XenServer installer: users are prompted to provide any extra drivers or supplemental packs at installation time.
  In addition, on upgrade, users are provided with a list of currently installed packs, and warned that they may require a new version of them that is compatible with the new version of XenServer.
- Flexibility in release cycles: partners are no longer tied to only releasing updates to their add-on software whenever new versions of XenServer are released.

Instead, partners are free to release as often as they choose.
The only constraint is the need to test packs on the newest version of XenServer when it is released.

- Integration with Server Status Reports: supplemental pack metadata can include lists of files (or commands to be run) to be included when a Server Status Report is collected using XenCenter. Pack authors can choose to create new categories, or add to existing ones, to provide more user-friendly bug reporting.
- Guarantee of integrity: supplemental packs are signed by the creator, allowing users to be certain of their origin.
- Include formal dependency information: pack metadata can detail installation requirements such as which versions of XenServer the pack can be installed upon.
- Inclusion in the Citrix Ready catalogue: partners whose supplemental packs meet certain certification criteria will be allowed to list their packs in the Citrix Ready online catalogue, thus increasing their visibility in the marketplace.
  Note that partners must become members of the program before their packs can be listed: the entry level category of membership is fee-free.

What or what not to include in a supplemental pack?

Citrix recognises that partner organizations can contribute significant value to the XenServer product by building solutions upon it.
Examples include host management and monitoring tools, backup utilities, and device-specific firmware.
In many cases, *some* of these solutions will need to be hosted in the XenServer control domain, dom0, generally because they need privileged access to the hardware.

Whilst supplemental packs provide the mechanism for installing components into dom0, try to install *as little as possible* using packs.
Instead, place the majority of partner software into appliance virtual machines, which have the advantage that the operating system environment can be configured exactly as required by the software to be run in them.

The reasons for this stipulation are:

- XenServer stability and QA: Citrix invests considerable resources in testing the stability of XenServer.
  Significant modifications to dom0 are likely to have unpredictable effects on the performance of the product, particularly if they are resource-hungry.

- Supportability: the XenServer control domain is well-known to Citrix support teams.
  If it is heavily modified, dom0 becomes very difficult to identify whether the cause of the problem is a component of XenServer, or due to a supplemental pack.
  In many cases, customers may be asked to reproduce the problem on an unmodified version of XenServer, which can cause customer dissatisfaction with the organization whose pack has been installed.
  Similarly, when a pack author is asked to debug a problem perceived to be with their pack, having the majority of the components of the pack in an appliance VM of known/static configuration can significantly ease diagnosis.
- Resource starvation: dom0 is limited in memory and processing power.
  If resource-hungry processes are installed by a supplemental pack, resource starvation can occur.
  This can impact both XenServer stability and the correct functioning of the supplemental pack.
  Note that Citrix does not advise increasing the number of dom0 vCPUs.
- Security: XenServer dom0 is designed to ensure the security of the hosts that it is installed on to.
  Any security issues found in software that is installed into dom0 can mean that the host is open to compromise.
  Hence, the smaller the quantity of software installed into dom0 by a pack, the lower the likelihood that XenServer hosts will be compromised due to a flaw in the software of the pack.

Partners often ask whether supplemental packs can include heavy-weight software, such as the Java runtime environment, or a web server.
This type of component is not suitable for inclusion in dom0. Instead place it in an appliance VM.
In many cases, the functionality that is desired can be achieved using such an appliance VM, in conjunction with the XenServer Management API (Xen API).
Citrix can provide advice to partners in such cases.

Getting started

November 21, 2023

This chapter describes how to setup a base XenServer system, running a DDK Virtual Machine (VM), for examining the examples provided in this document, and for use in the development of supplemental packs.

If you want to construct supplemental packs as part of your own build systems, consult the appropriate section later in this document.

The high-level process of setting up a DDK VM to create a supplemental pack is:

1. Obtain matching XenServer product and DDK build ISOs.
2. Install XenServer onto a host server.
3. Install the XenCenter administrator console onto a Windows-based machine.
4. Use XenCenter to import the DDK onto the XenServer host as a new virtual machine.

Installing XenServer

Installing XenServer only requires booting from the CD-ROM image, and answering a few basic questions.
After setup completes, take note of the host IP address shown, as it is required for connection from XenCenter.

Installing XenCenter

XenCenter, the XenServer administration console, must be installed on a separate Windows-based machine.
Inserting the XenServer installation CD will run the XenCenter installer automatically.
Once installed, the XenCenter console will be displayed with no servers connected.

Connect XenCenter to the XenServer host

Within XenCenter select the **Server > Add** menu option and supply the appropriate host name/IP address and password for the XenServer host.

Select the newly connected host in the left-hand tree view.

Importing the DDK VM through XenCenter

> **Note:**
>
> You can also import the DDK directly on the host using the **xe** Command Line Interface (CLI).

1. Download the XenServer DDK from the XenServer downloads page onto the system where XenCenter is installed.
2. On the **File** menu, select the **Import** option. The Import Wizard is displayed.
3. Click **Browse** and select DDK file that you downloaded.
4. Proceed through the Import Wizard to select the server, storage, and network for your DDK VM.
5. Select **Start the new VM automatically as soon as the import is complete**.

6. Finish the Import Wizard.

The DDK VM starts automatically.

Importing the DDK VM using the CLI

The DDK VM can also be imported directly on the XenServer host using the xe CLI and standard Linux commands to mount the DDK ISO.

- Mount the DDK ISO and import the DDK VM:

```
mkdir -p /mnt/tmp
mount <path_to_DDK_ISO>/ddk.iso /mnt/tmp – o loop,ro
xe vm-import filename=/mnt/tmp/ddk/ova.xml
<!--NeedCopy-->
```

  The universally unique identifier (UUID) of the DDK VM is returned when the import completes.

- Add a virtual network interface to the DDK VM:

```
xe network-list
<!--NeedCopy-->
```

  Note the UUID of the appropriate network to use with the DDK VM, typically this will be the network with a `name-label` of `Pool-wide network associated with eth0`.

```
xe vif-create network-uuid=<network_uuid> vm-uuid=<ddk_vm_uuid>
    device=0
<!--NeedCopy-->
```

> **Note:**
>
> Use tab completion to avoid entering more than the first couple of characters of the UUIDs.

- Start the DDK VM:

```
xe vm-start uuid=<ddk_vm_uuid>
<!--NeedCopy-->
```

Using the DDK VM

Select the DDK VM in the left pane and then select the Console tab in the right pane to display the console of the DDK VM to provide a terminal window in which you can work.

The DDK VM is Linux-based so you are free to use other methods such as ssh to access the DDK VM. You can also access the DDK VM console directly from the host console.

Adding Extra Packages to the DDK VM

The DDK is built to be as close as possible to the XenServer control domain (Dom0).
This means that only a small number of extra packages are present in the DDK (to enable the compilation of kernel modules) as compared to Dom0.
In some cases, partners who want to use the DDK as a build environment might want to add extra packages (e.g. NIS authentication) to the DDK.

Because the DDK (and Dom0) are based on CentOS, any package that is available for that distribution can be installed into the DDK, using the Yum package manager.
However, it is necessary to explicitly enable the CentOS repositories to allow such installation.
Package installation must therefore be carried out using the command:

```
yum install <packageName>
```

Accessing the DDK VM console from the host console

The DDK VM text console can be accessed directly from the XenServer host console instead of using XenCenter.
Note that using this method disables access to the DDK console from XenCenter.

- While the DDK VM is shut down, disable VNC access on the VM record:

  ```
  xe vm-param-set uuid=<ddk_vm_uuid> other-config:disable_pv_vnc=1
  ```

- Start the VM

  ```
  xe vm-start uuid=<ddk_vm_uuid>
  ```

- Retrieve the underlying domain ID of the VM:

  ```
  xe vm-list params=dom-id uuid=<ddk_vm_uuid> --minimal
  ```

- Connect to the VM text console:

  ```
  /usr/lib/xen/bin/xenconsole <dom-id>
  ```

- When complete, exit the xenconsole session using **CTRL-]**

For more information about using the xe CLI, see the command line interface documentation.

Building the example packs

November 16, 2023

To make the process of building an Update package as easy as possible, we have included a number of examples in the Driver Development Kit (DDK), which Citrix makes available to our partners with each build.

Import the DDK onto the XenServer host.
Refer to the instructions described in `/root/examples/README.txt`, which outlines the example scripts for generating GPG test keys as well as the various configuration options for generating a pack.

For most cases, copying the example build files and customizing them with pack specific information is sufficient.

Pack UUIDs: Each pack has a UUID included in the metadata, which is required to build an update package.
The UUID must be unique to the update being generating:

- The same UUID cannot refer to different updates.
- The same update cannot have multiple UUIDs.

A number of examples are supplied in the DDK under `/root/examples`.
These include:

- **Userspace**: a simple example of a pack containing only programs and files that relate to a userspace.
- **Driver**: a simple kernel driver.
- **Combined**: an example which contains kernel and userspace files.

There are specific rules for packaging kernel device drivers.
For more information, see Rules and guidelines.

Within each directory there is a:

- **Source tree**: a directory containing a collection of files.
- **Specification file**: a file that describes how to build an RPM.
- **Makefile**: a file used to automate the creation of a supplemental pack.

To build a specific example, use the following commands:

```
cd /root/example/<dir>
make
```

This will result in the following files being created:

- ***<pack>*.iso** - the supplemental pack CD image.

Where <*pack*> is the name of the pack.

Test Signing Key Generation

When the first example pack is built, a new GnuPG key pair is created. You will be prompted for a passphrase that must be entered whenever the private key is used to sign a pack. This key pair is only intended for developer testing. For information on generating a key pair to use for released supplemental packs, see The GNU Privacy Handbook.

To allow partners to release software that is installable in Dom0, Citrix requires partners provide us with the public key corresponding to their GPG key pair generated with the following requirements:

- ASD Type = RSA
- Bits = 2048
- Expiry date = preferably none, else >10 years.
- No support for subkeys as RPM does not handle this properly.
- Naming convention: RPM-GPG-KEY-<VENDOR>

Installing the Test Key

Before a pack that has been signed with a test key can be installed on any XenServer hosts, the public key must be imported into the host on which the pack is installed.

1. Copy the public key from the DDK VM to the host using the following command:

   ```
   DDK# scp /root/RPM-GPG-KEY-DDK-Test root@CitrixHypervisor:
   ```

2. Import the key on the host using the following command:

   ```
   XS# /opt/xensource/debug/import-update-key RPM-GPG-KEY-DDK-Test
   ```

> **Note:**
>
> To allow developer testing, a script is now included in Dom0 that enables our partners to import an update key manually:
>
> ```
> /opt/xensource/debug/import-update-key <PATH-TO-KEY-FILE>
> <!--NeedCopy-->
> ```

Producing driver RPMs

May 15, 2023

In order to produce a supplemental pack that contains kernel modules (drivers), the DDK must be used to compile the driver(s) from their source code, against the XenServer kernel. This chapter describes the process.

Directory structure

Although the examples located in the `/root/examples/` directory contain various subdirectories, in practice, most supplemental pack authors will not use this structure.
The following example considers a supplemental pack that contains both kernel modules and user-space components (as the `combined` example does).

In the `combined` case, two RPMs will be created, one containing the kernel modules, and the other the "data"or userspace portions (configuration files, firmware, modprobe rules).
Hence, two specification files are present, which specify the contents of each RPM that is to be created.

Place the kernel driver source code in the `/root/rpmbuild/SOURCES/` directory as a tar archive (it can be gzipped or bzip2 compressed), whose name is of the format `<module-name>-<module-version>`.
Meanwhile, the specification files for the RPMs to be created will be stored in a directory elsewhere. we recommend authors make a copy of the specification files and Makefile found in the `examples/combined/` directory as a starting point.
The Makefile contains various useful build targets that can be adjusted for the kernel module sources being used.

Place all non-kernel module components in a directory named `<module-name>-data-<version>`, for example, `helloworld-data-1.0`.
The corresponding specification file would be `helloworld-data.spec`.
It is suggested that this subdirectory be placed in the same directory as the specification files.

Makefile variables

The Makefile includes several metadata attributes which must be customized according to the contents of the pack. These are as follows:

- **SPEC**: the specification file for the driver RPM.

- **DATA_SPEC**: the specification file for the userspace components RPM.
- **LABEL**: by default, taken from the "Name:"field of the driver RPM specification file, but can be edited if so desired.
- **TEXT**: a free text field describing the function of the driver. This is displayed on installation.
- **UUID**: a universal unique identifier generated by the UUIDgen command.
  Every pack must have a different UUID and a given pack must have the same UUID each time it is built.
- **PACK_VERSION**: the version number of the pack (this defaults to the build of the DDK being used, but can be changed by pack authors).
- **PACK_BUILD**: the build number of the pack (this defaults to the build of the DDK being used, but can be changed by pack authors).
- **RPM_VERSION**: the version number to be used for the kernel modules RPM. Citrix advises authors to set this to the version of the driver source being used.
- **RPM_RELEASE**: the release number of this version of the RPM. For example, the same version of driver might be re-released in a supplemental pack, and hence need a new release number.
- **DATA_RPM_VERSION**: the version number to be used for the userspace RPM.
- **DATA_RPM_RELEASE**: the release number to be used for the userspace RPM.

Creating the kernel module specification file

The kernel module packages are built according to the instructions in the specification file.
The following sections of the specification file affect the building of a package. Ensure you set them to appropriate values:

- **Name**: a unique ID, preferably the name of the kernel module.
- **Source**: the exact filename (without the path) of the tar archive that contains the sources for the kernel module, expected to be of the form `<module-name>-<module-version>`.
- **Summary**: a short description of the driver.
- **%files** is a list of files that are to be compiled into the RPM.
  Kernel modules must be located in a directory named `extra` (within `/lib/modules/<kernel-version>/`).
- **%changelog** describes changes that have been made to the driver.

Building the modules

Each kernel module RPM must be built against against the XenServer kernel headers.
The example Makefile provides a `build-rpms` target that automates the build.
The userspace RPM is also built if necessary.
The RPMs are output into the `/root/rpmbuild/RPMS/x86_64` directory.

If the RPM does not build, it is important that the following be checked:

- The `Source` parameter of the kernel RPM specification file *must* be the filename of the compressed tar archive containing the source code for the module, located in `/root/rpmbuild/SOURCES`.
- The `%prep` section of the example specification file relies on the compressed tar archive, when expanded, creating a directory named `<module-name>-<module-version>`. If this is not the case, (for example, if it creates a directory named `<module-version>`), the `%setup -q -n` step can be amended to be, for example, `%setup -q -n %{ version }`.
- The `%build` section of the example specification file relies on the directory `/root/rpmbuild/SOURCES/<module-name>-<module-version>/` containing the Makefile or KMake file that will build the kernel module `<module-name>`.
  If this Makefile is in a subdirectory, the `%build` section will need a `cd <subdirectory-name>` step added to it, before the `%{ __make }` step. Similarly, you will need to add the same step into the `%install` section.
- If, for any reason, the Makefile included in the compressed tar archive needs to be heavily patched in order to work correctly with the DDK, Citrix suggests that a new version of the file be created, with the appropriate fixes, then a patch generated using the `diff` command.
  This patch can then be applied in the `%prep` section of the specification file, immediately following the `%setup` step.
- If the kernel module itself fails to compile, (rather than the RPMs failing to build), it may be that the source being used is incompatible with the kernel version that is used in XenServer.
  In this case, contact the author of the driver.

Including driver RPMs in supplemental packs

The next chapter details exactly how to include not only driver RPMs produced in the above manner in a supplemental pack, but also any other arbitrary RPMs.
If a pack is only to include driver RPMs plus some associated configuration or firmware, it can be produced directly by using the Makefile `$(ISO)` target, which runs the `build-update` script.
The script is run with the appropriate arguments to include the metadata that was given in the Makefile.

Format for releasing drivers

If a supplemental pack contains drivers, it must also be shipped with the source code to those drivers, to fulfill obligations under the GNU General Public License (GPL).
We recommend that pack authors create a zip file containing the pack ISO, a compressed archive of any relevant source code, and the MD5 checksum files that are associated with the pack metadata and the ISO, produced by `build-update`.

Releasing multiple drivers in a single pack-

Server hardware manufacturers might want to issue a single supplemental pack that contains multiple drivers.
Three options are available, depending on the desired result:

- Make individual copies of the `/root/examples/driver` directory, one for each driver. Then produce the two RPMs for each driver using the `build` target in the `Makefile`. Collect all the RPMs into one place, and then run `build-supplemental-pack.sh`.
- Create a specification file for each driver (similar to that in `/root/examples/driver`). Adjust the `Makefile` to compile all the drivers and produce RPMs for each one, and then run `build-supplemental-pack.sh`.

Creating a supplemental pack

November 6, 2023

Supplemental packs can be created containing existing RPMs provided they meet the requirements given below.
If standard packages that are not shipped in XenServer are to be included, these must be from the appropriate CentOS distribution that the XenServer dom0 is based upon.
In Citrix Hypervisor 8.0, this is CentOS 7.5. Alternatively, components can packaged as RPMs using a custom spec file.

If a pack will only contain drivers, it is normally known as a XenServer Driver Disk.
However, the mechanisms used to build and install a driver disk are the same as for any other supplemental pack.
One key point is that for drivers, the source code must be provided to the DDK, in order that the drivers be compiled for the correct kernels.
For other pack components, no such compilation is necessary.

Syntax of build-update

All supplemental packs are constructed using the `/usr/bin/build-update` script.
This provides a simple way to provide metadata about the pack, by taking a number of options and arguments.
The significance of each one is discussed in the sections that follow.

Apart from the metadata switches, any further arguments to `build-update` are taken to be names of the RPMs to be included in the supplemental pack.

Name, vendor, and version information

Basic details concerning the pack authoring organization, name, and version number must be provided. The relevant switches are:

- `--uuid`: a universal unique identifier generated by the **uuidgen** command.
- `--label`: an identifier that identifies the pack.
- `--text`: a free text string describing the pack (enclosed by double quote marks).
- `--version`: the pack version (likely to be of the form 1.2.3).

Declaring Pack dependencies

Supplemental Packs must describe the version of XenServer to which they are compatible. This is done using the `--base-requires` switch:

```
--base-requires “ product-version=x.x.x ”
```

A brief example

As an illustration of how `build-update` works, pack authors can create an example pack by placing all constituent RPMs in a directory:

```
mkdir packages
cd packages
cp /root/rpmbuild/RPMS/x86_64/helloworld-1.0-1.x86_64.rpm .
```

The pack metadata is then created, using the script.

```
build-update --uuid 3fbce6cf-5cd2-4d32-9602-e8122c562169 --label
    example pack --version 1.0.0 \
--description "An Example Pack" --base-requires "product version=8.0.0"
    \
--key "Example Updates (update) <example@example.com>" --keyfile /root/
    RPM-GPG-KEY-XS-DDK-TEST \
-o example.iso *.rpm
```

A CD image is then made from the contents of this directory.

Adding files to Server Status Reports

XenServer provides a convenient mechanism for users to collect a variety of debugging information when opening a support case, known as a Server Status Report in XenCenter, or `xen-bugtool` on the CLI.
To aid partners in supporting their supplemental packs, it is recommended that pack authors add to the list of files collected as part of these Status Reports, using the method outlined below.

Server Status Reports can include not only files, such as logs, but also the outputs of any normal scripts or commands that are run in the control domain (dom0).
For convenience, the items collected as part of a Report are divided into categories.
For example, the Network Configuration category collects the output of tools such as `ifconfig`, as well as network configuration files.
we recommend that pack authors create new categories if appropriate, but also consider adding to existing categories.

As an example, partner Acmesoft, who produce software that manages the configuration of a special network card, might want to create a category called Acmesoft, under which various Acmesoft-specific log files are collected.
However, they might also want to add (other) files related to networking to the Network Configuration category, as it makes most sense from a user perspective that they be collected as part of this category.

Each category has a level of confidentiality attached to it, which expresses how much personally identifiable information (PII) might be present in the files that are collected as part of that capability.
This ensures that users are made aware before they send Status Reports to support teams of what they might be disclosing.
There are four levels of confidentiality: no PII, possibly some PII if the file to be collected has been customized, possibly contains some PII, and definitely contains
PII.

Extending an existing category

To extend an existing category, create an XML file in `/etc/xensource/bugtool/<category>/` directory (where *<category>* is the category name).
Three elements within an outer `<collect>` element will be supported:

- `files`: a list of one or more files (separated by spaces, hence no spaces are allowed within the file names).
- `directory`: a directory to be collected, with (optionally) a pattern that will be used to filter objects within.
  The pattern must be a valid Python regular expression.
  The `negate` attribute allows the sense of the pattern to be inverted: this attribute is provided for ease of readability purposes, as negation could also be included
  in the regular expression itself.
- `command`: a command to be run and its output collected.
  The optional **`label`** attribute specifies a name for the output file.
  If this is not specified, the command name will be used.

For example:

```
    <collect>
      <files>file1 file2</files>
      <directory pattern=".*\.txt$" negate="false">dir</directory>
      <command label="label">cmd</command>
    </collect>
<!--NeedCopy-->
```

> **Note:**
>
> When extending existing categories, any files added must have the same (or lower) confidentiality levels as the category in question.

Existing categories are:

Category name	Description	PII level	Collected by default?
CVSM	Citrix StorageLink configuration	No	Yes
disk-info	Disk information	Maybe	Yes
firstboot	First-boot scripts	Yes	Yes
hardware-info	Hardware information	Maybe	Yes
high-availability	High availability	Maybe	Yes
host-crashdump-dumps	Crash dump files	Yes	No
host-crashdump-logs	Crash dump logs	No	No
kernel-info	Kernel information	Maybe	Yes

Category name	Description	PII level	Collected by default?
loopback-devices	Loopback devices	Maybe	Yes
multipath	Multipathing configuration	Maybe	Yes
network-status	Network scripts	If customized	Yes
pam	Authentication module configuration	No	Yes
process-list	Process listing	Yes	Yes
persistent-stats	Persistent statistics	Maybe	Yes
system-logs	System logs	Maybe	Yes
system-services	System services	No	Yes
tapdisk-logs	Storage subsystem logs	No	No
vncterm	VNCTerm crash dumps	Maybe	No
wlb	Workload Balancing status	No	Yes
XYes1	X server logs	No	Yes
XYes1-auth	X11 authority	No	Yes
xapi-subprocess	XenServer daemon subprocesses	No	Yes
XenServer-config	XenServer configuration	Maybe	Yes
XenServer-domains	XenServer domains list	No	Yes
XenServer-databases	XenServer database	Yes	Yes
XenServer-install	XenServer installation log files	Maybe	Yes
XenServer-logs	XenServer logs	Maybe	Yes
xen-info	Hypervisor configuration	Maybe	Yes
xha-liveset	High availability liveset	Maybe	Yes
yum	RPM package database	If customized	Yes

A number of other categories exist. These categories are purely for development and test purposes.

Do not extend them in your supplemental packs.

Adding new categories

To add a new category, create an XML file with the name `/etc/xensource/bugtool/<category>.xml` (where <*category*> is the new category name).
Include a single `<capability>` element, with the following optional attributes:

- `pii`: the degree of confidentiality that is inherent in the files to be collected (personally identifiable information).
  This must be set to one of `no` (no PII), `if_customized` (PII would only be present if the file has been customized by the user), `maybe` (PII may be present in this file), or `yes` (there is a very high likelihood of PII being present).
- `min_size`: the minimum size (in bytes) of the data collected by this category.
- `max_size`: the maximum size (in bytes) of the data collected by this category.
- `min_time`: the minimum time (in seconds) that the commands are expected to take to run to completion.
- `max_time`: the maximum time (in seconds) that the commands are expected to take to run to completion.
- `mime`: the MIME type of the data to be collected. This must be one of `application/data` or `text/plain`.
- `checked`: specifies whether this category is to be collected by default (set to **`true`**) or not (set to **`false`**).
- `hidden`: when set to **`true`**, this category will only be collected if explicitly requested on the CLI. It will not be visible in XenCenter.

> **Note:**
>
> If the data to be collected by a capability exceeds the constraints placed upon it (that is, the maximum size of data to be collected is higher than `max_size`), then the data is *not* collected. The `min_size` and `min_time` attributes are purely for user information: if the amount of data collected, or time taken for its collection, is less than these attributes, the data *is* collected.

Specify the data to be collected as part of this category by using one or more XML files located in the `/etc/xensource/bugtool/<category>/` directory, as described in the previous section.

At present, XenCenter displays any new categories that are added by supplemental packs, but does not provide a mechanism for pack authors to give descriptions of them in the Server Status Report dialogue.
Ensure that any new categories have suitably descriptive names.

Automating pack installation at XenServer installation time

If a partner has obtained the necessary agreement from Citrix to distribute XenServer, it is possible to create a modified installation ISO that contains the XenServer installation files, plus one or more supplemental packs.
This allows a partner to distribute a single ISO that can seamlessly install XenServer and the supplemental pack(s).
There are two steps to this process: the first involves combining the installation ISO with the pack ISO; the second requires an answerfile to be created.

Including an answerfile on the XenServer installation ISO

Answerfiles allow the responses to all the questions posed by the XenServer installer to be specified in an XML file, rather than needing to run the installer in interactive mode.
The XenServer DDK includes a script to add an answerfile to the XenServer main installation ISO, to allow installation to be carried out in an unattended fashion.
The `/usr/local/bin/rebuild-iso.sh` script can be used within the DDK (though the default root disk size within the DDK is unlikely to be sufficient to perform the necessary ISO re-packing operations, and hence network storage is recommended).
Alternatively, the script, plus `/usr/local/bin/rebuild.functions`, can be copied to an external machine that has `mkisofs` installed, and used there.

The `rebuild-iso.sh` script takes in the XenServer installation ISO, plus the files to be added to it, and outputs a combined ISO. Its syntax is:

```
rebuild-iso.sh
       [--answerfile=<answerfile>]
       \[--include=<file>|<directory>]
       [--label=<ISOLabel>]
       inputFile.iso outputFile.iso
```

The <*answerfile*> must be a valid XenServer automated installation file.
Details of the syntax of this file are given in the relevant section of the XenServer Product Documentation.
As part of an answerfile, it is possible to specify scripts that may be run directly after the installation completes.
Whilst such scripts can be on a web server, it is also possible to place them on the ISO, with the answerfile.
The `--include` switch allows such files to be added to the ISO output by `rebuild-iso.sh`. Finally, the `--label` switch controls the volume label of the resulting ISO.
If no label is specified, the label of the input ISO is used.

It is worth noting that whilst an answerfile specifies the answers to installation questions such as keyboard layout and target disks, it does not specify which repositories to install from. This is because for an automated installation, all repositories specified in the `XS-REPOSITORY-LIST` file are installed. Therefore, provided that all supplemental packs that are to be installed are included in the `XS-REPOSITORY-LIST` file, they will be installed automatically directly following the installation of XenServer itself.

Rules and guidelines

January 25, 2024

Kernel modules

Kernel modules must be built and packaged according to the following:

- Modules must be placed in an RPM.
- All modules must be located under the directory `/lib/modules/<kernel>/updates` where <*kernel*> is the version of the kernel.

To ensure a pack is fully conformant, we recommend basing it on one of the examples in the DDK.

> **Note:**
>
> The XenServer build into which a kernel module (driver) is installed *must* be the identical build to the DDK that was used to build the pack in which the driver is contained.
> If it is not, the resulting driver disk will not install on XenServer.

Post-install scripts

Provided they comply with the following constraints, RPMs may contain scripts that are invoked during installation. Such scripts might be necessary in order to add appropriate firewall rules, or log rotation configuration, that is specific to the pack.

1. Scripts must not start processes.

2. Scripts must not assume that XenServer is booted and running (as it may be that the pack is being installed as part of an initial XenServer installation.
3. In the light of the previous point, if firewall rules are to be added using a post-install script, the script *must* execute `iptables-restore < /etc/sysconfig/iptables` *before* adding its own rules (using `iptables -A`), and then run `iptables-save > /etc/sysconfig/iptables`. This ensures that the default rules are loaded before the collection of existing and new rules are saved. Failure to save the existing rules will mean that when a pack is installed as part of a XenServer host installation, the default XenServer firewall rules will be lost.

If an RPM needs to distinguish between a running host and an installation environment, the following code fragment may be used:

```
if runlevel >/dev/null 2>&1; then
# running host
else
# installation
fi
```

The following types of file *must* be placed in the appropriate directories:

- Udev rules: must be located in `/etc/udev/rules.d/`.
- Firmware: must be located in `/lib/firmware/updates/<kernel>`.
- Configuration details: must be located in `/etc/`.
- Documentation: must be located in `/usr/share/doc/`
- Firewall rules: must be added using `iptables -A`, having *first* run `iptables-save > /etc/sysconfig/iptables` (see above).

Handling upgrades

On upgrade, pack authors might want to transfer configuration or state information from the previous installation: this section describes how such a transfer may be achieved.

Pack upgrade during a XenServer upgrade

When a XenServer host is upgraded, the installer replaces the file system before supplemental packs are installed.
This affects the way individual RPMs interact with upgrades.
To enable configuration files (or other configuration data, such as databases) to be carried over, the installer makes the file system of the previous installation (which is automatically backed-up to another partition) available to the RPM scripts.
This is done through the `XS_PREVIOUS_INSTALLATION` environment variable.

Therefore, in order to migrate state across upgrades, supplemental pack authors must create a suitable script that runs as part of an RPM installation, and migrates the state. Specifically, the migration is from the old root file system pointed to by `XS_PREVIOUS_INSTALLATION` to the new file system mounted on `/`.

For an example of how this can be done, see the `%post` script in `examples/userspace/helloworld-user.spec`.

Pack upgrade on an existing XenServer installation

It is expected that on each release of XenServer, supplemental pack authors are likely to release new versions of their packs. However, if a pack author releases an update between XenServer releases, existing installations of the old version of the pack would need to be upgraded. This upgrade path is the responsibility of the pack author, as the location of the configuration data of the old version is on the root file system, wherever the RPMs installed it to. The update installation process upgrades all RPMs contained within the pack, hence these RPMs must be aware of how to deal with the existence of any relevant configuration files.

Uninstallation

Supplemental Packs that only contain userspace packages may include a script that removes the pack from a host.

- Uninstalls the packages
- Uninstalls other packages used to apply the pack
- Causes xapi to remove the pack from the database

Building packs in existing build environments

Citrix recognises that many partners have existing build systems that are used to produce the software that might be integrated into a supplemental pack. To facilitate this, pack authors are *not* required to make use of the Driver Development Kit VM, *if* they are producing packs that do *not* contain drivers (as these need to be compiled for the correct XenServer kernels).

If a pack author wants to distribute drivers as part of a supplemental pack, (or a pack consisting solely of drivers, commonly known as a Driver Disk), then the driver(s) will need to be compiled using the DDK. However, there is no barrier to pack authors including the driver disks that are output by the DDK in their own build processes. Citrix does not support the compilation of drivers for XenServer in any way other than using the DDK VM.

To build a supplemental pack (but not a driver) as part of an existing build process, the only package that is necessary from the Binary Packages ISO is `update-package`.

These can be used in any environment to create an appropriate pack. Note, however, that this environment must contain various tools that are normally found in standard Linux distributions, including `tar`, `mkisofs`, `sed`, and `rpm`.

> **Warning**
>
> When integrating these scripts as part of another build system, bear in mind that Citrix may update these scripts as new versions of XenServer are released.
> Ensure that you update this package from the new version of the DDK before building packs for the new version of XenServer.

For pack authors who want to produce a supplemental pack as an output of another build system, but who want to include drivers, follow this procedure:

1. Copy the driver source into a running DDK VM the corresponds to the build of XenServer that the drivers will be targeted at.
2. Produce driver *RPMs* (rather than a supplemental pack ISO). This can be achieved using the `$(RPM-FILE)` rule in the standard `Makefile` provided in the example packs.

3. By default, this command will output three driver RPMs into `/root/rpmbuild/RPMS/x86-64`. Ignore the `debuginfo` RPM, and take the other RPMs for inclusion in their supplemental pack. These RPMs can be treated in the same way as any other RPM to be shipped in the pack.
4. Provide these RPMs to the non-DDK build system, for inclusion in the finished pack. Evidently, this process assumes that pack authors will be releasing new versions of their packs more frequently than the XenServer kernel is changed, or that introducing new versions of RPMs into the alternative build system is more acceptable than introducing the DDK as the build system.

Packaging driver firmware

It is increasingly common for hardware manufacturers to produce components that load up-to-date firmware from the operating system that is running on the host machine. XenServer supports this mechanism, and firmware packages are installed to `/lib/firmware/updates/<kernel>`. Package the firmware in an RPM and include it as part of a supplemental pack.

Versioning

Supplemental pack versioning

Authors of supplemental packs are free to use whatever version numbering scheme they feel is appropriate for their pack. Given that many packs are likely to include RPMs of existing software, it is suggested that the pack version number correspond to the version of the software it contains. For example, if an existing management console RPM is at version 5, it is likely to be less confusing if the first version of the supplemental pack that contains this RPM is also version 5.

There is no reason why, if it is simple enough, a pack can't be suitable for multiple releases of XenServer *provided* that it does not depend on particular versions of other tools. In practice, we recommend that you re-release a pack for each new version of XenServer. Ensure that the pack is also fully tested by the authors on that new release. If a pack author chooses to release one pack for multiple XenServer releases, they must ensure that the dependency information expressed in the metadata of the pack uses the ge comparator for the XenServer product, where the version to be compared against is the lowest supported version of XenServer.

Kernel module versioning

Each kernel module is built against a specific kernel version. This kernel version is included in the RPM name to enable multiple instances to be installed. The version fields of a kernel module RPM are used to track changes to a driver for a single kernel version. Each time a driver is released for a particular kernel, the RPM version must be increased (as would be expected, given that there will have been changes made to the driver sources).

For example:

```
helloworld-modules-xen-2.6.18-128.1.6.el5.xs5.5.0.502.1014-1.0-1.i386.
    rpm
driver name = helloworld
kernel flavour = xen
kernel version = 2.6.18-128.1.6.el5.xs5.5.0.502.1014
RPM version = 1.0
RPM build = 1
```

XenServer Kernel version	Kernel module RPM version	Event
2.6.18.128.1.5…	1.0-1	Initial release of pack
2.6.18.128.1.5…	1.1-1	Driver bug fix
3.0.0…	2.0-1	New XenServer kernel, and new driver version

Therefore, if a supplemental pack contains a driver, it will be necessary to rebuild that driver for each update and major release of XenServer. Note that hotfixes do not normally change the kernel version, and hence the same driver can be used until a XenServer update ("service pack") is released.

As a general rule, if a pack contains only a single driver, it is strongly recommended that the version numbering of the pack be the same as that of the driver.

Packages compiled by, but not in, XenServer

Some of the packages that are included in the XenServer control domain are taken directly from the base Linux distribution, whilst others are modified and re-compiled by Citrix. In some cases, certain source RPMs, when compiled, result in more than one binary RPM. There exist a variety of packages where XenServer includes some, but not all, of the resulting binaries; for example, the `net-snmp` package results in the binary packages `net-snmp` and `net-snmp-utils`, but `net-snmp-utils` is not included in dom0.

If a supplemental pack author wants to include a binary package that falls into this category, that binary package will need to have the correct build number for the version of XenServer it is to be installed upon. Because Citrix re-compiles these packages, their build numbers will have a XenServer-specific build number extension. Therefore, pack authors will need to obtain these binary RPMs from Citrix.

To enable this process to be as simple as possible, Citrix produces an extra ISO (`binpkg.iso`) for each release of XenServer that contains all the packages that fall into this category. Contact Citrix to obtain this ISO.

Requirements for submission of drivers for inclusion in XenServer

Citrix encourages hardware vendors to submit any driver disks released for XenServer to Citrix in order that the drivers may be incorporated into the next release of the product. In order to make this process as simple as possible, vendors are requested to take note of the following requirements:

1. Any driver submitted must include its full source code, that is available under an open source license compatible with the GNU General Public License (GPL).
2. Any binary firmware submitted must either be already publicly available under a license allowing re-distribution, or the vendor must have a current re-distribution agreement with Citrix.
3. If a new, (rather than an update to an existing) driver is being submitted, Citrix will review it in order to confirm that it is compatible with the current support statements made concerning XenServer. It may be that a driver is rejected because it is

monolithic, or enables a feature which is not currently officially supported. This may also be the case with radical changes made to drivers that are already in the product. Partners who consider that their driver(s) fall into this category must contact Citrix as early as possible, in order that both organizations'engineering teams are able to ascertain how to proceed.

4. Strict time limits apply to submissions (see below). Vendors must make their drivers available to Citrix as soon as possible, rather than waiting until these deadlines, as testing may result in fixes being required, which then need to be integrated into the product.
5. Certain drivers are deemed critical to automated testing by Citrix of XenServer. Any (entire-driver) updates to these drivers must be provided a minimum of 6 weeks prior to the beta RTM date of the release in which they are to be included. Partners whose drivers are on this list will be informed of this constraint.
6. All other entire-driver updates (or new drivers) must be provided to Citrix a minimum of 5 weeks prior to beta RTM.
7. Any updates to drivers (for example, no new drivers) received after these dates must be in the form of small, targeted fixes for specific issues. Patches must be submitted that can be understood by a reasonably experienced person, together with a description of the flaw that particular patch addresses. Provide each fix in the form of a separate patch, with an indication of what the flaw fixed is, the effects the flaw would have if it is not addressed, and whether such issues have already been seen by customers. Significant additions of functionality, or very large patches will not be accepted.
8. Close to the RTM date of the beta of the release concerned, it is unlikely that patches will be inserted into the beta release (though they may be incorporated into the final release, if they are judged to be of sufficient importance). Only in exceptional circumstances will patches received fewer than 2 weeks prior to beta RTM be incorporated into the beta. The preferred target for all driver updates is the beta release, in order to achieve maximum testing benefit.
9. Submitting a GPG Key: Once the key has been generated, when creating the update pack, the name of the GPG key is baked into the Yum repo

metadata. This means the public key file cannot be renamed without resigning the Update package.

When the key-pair has been generated, export the public part (using ASCI Armor) and create a ticket on Citrix Issue Tracker to include it in the inbox.

Does XenServer already include a driver for my device?

XenServer includes a wide variety of drivers, including many that are distributed (inbox) with the kernel that dom0 is based upon. It may therefore be the case that XenServer includes a driver that enables a device that is not present on the XenServer Hardware Compatibility List (HCL). This is particularly the case where a device is sold by multiple companies, each of which refers to it with a different name.

Because each driver included in XenServer includes information concerning which PCI device IDs it claims, the simplest way to ascertain whether a device is supported is to first find its device ID.

If the device is present in a running Linux-based system, the `lspci -v` command can be used, which will provide output which includes the information on all devices present in the host. If the `-n` switch is given, numeric IDs will be provided.

If only the name of the device is known, use the PCI ID database (https://pci-ids.ucw.cz/) to ascertain what the ID of the device is. This database will also provide alternate names for the device, which may of use if the exact name is not listed in the XenServer HCL.

If the an alternate name for the device is not found on the HCL, then either the device has not been tested on XenServer, or a driver for it is not included in XenServer. To confirm whether a suitable driver is included, consult the list of PCI IDs the XenServer kernel supports, found in `/lib/modules/<version>/modules.pcimap`.

Testing and certification

April 8, 2024

XenServer uses a modified Linux kernel that is similar but not identical to the kernel distributed by a popular Linux distribution. In contrast, the XenServer control domain is currently based on a different distribution. In addition, the 32 bit XenServer control domain kernel is running above the 80K lines of code that are the 64 bit Xen hypervisor itself. While Citrix is very confident in the stability of the hypervisor, its presence represents a different software installation than exists with the stock vendor kernel installed on bare hardware.

In particular, there are issues that may be taken for granted on an x86 processor, such as the difference between physical and device bus memory addresses (for example **virt_to_phys()** as opposed to **virt_to_bus()**), timing, and interrupt delivery which may have subtle differences in a hypervisor environment.

For these reasons, expose hardware drivers to a set of testing on the XenServer control domain kernel to ensure the same level of confidence that exists for drivers in enterprise distribution releases. Similarly, userspace software that is included in supplemental packs must be tested comprehensively, to ensure that the assumptions it makes about the environment in which it runs (for example, concerning the presence of certain executables) are not invalidated.

The remainder of this section considers driver testing. If you want to release supplemental packs that do not only contain drivers, contact Citrix for advice. As a minimum, such pack authors must comprehensively test the functionality of the software being included
in the pack, as well as perform stress testing of the XenServer major features, to ensure that none are impacted by the software in the pack.

Testing scope

Assuming the driver in question has already undergone verification testing on a Linux distribution very similar to the one used in the XenServer control domain, a subset of the verification test suite with a focus on representative tests is typically sufficient.

Some common areas of focus are:

- *Installation verification.* Installation is now performed using an RPM, which may be different from how the driver is typically installed.

- *CLI operation.* If a CLI is included, its operation is a key scenario and typically provides a good end-to-end exercising of all related code.
- *Adapter configuration, non-volatile/flash RAM, or BIOS upgrades.* Any functions that access the physical hardware must be verified due to the presence of the Xen hypervisor and its role in coordination of hardware resources amongst virtual machines.
- *Data integrity and fault injection.* Long-haul and/or stress-based data integrity verification tests to verify no data corruption issues exist. Basic fault injection tests such as cable un-plug/re-plug and timeout handling for verification of common error conditions.
- *Coverage of a representative set of device models.* For example, if a Fibre Channel HBA driver supports a set of models that operate at either 2 Gb/s or 4 Gb/s, include one model each from the 2 Gb/s and 4 Gb/s families for testing.
- *Key hardware configuration variations.* Run any hardware configurations that exercise the driver or related code in significantly different ways, such as locally versus remotely attached storage.

Running tests

Since the physical device drivers run in the XenServer control domain, the majority of tests will also be run in the control domain. This allows simple re-use of existing Linux-based tests.

To provide high-performance device I/O to guest domains, XenServer includes synthetic device drivers for storage and networking that run in a guest and communicate their I/O requests with corresponding back-end drivers running in the control domain. The back-end drivers then issue the I/O requests to the physical drivers and devices and manage transmitting results and completion notifications to the synthetic drivers. This approach provides near bare-metal performance.

As a result, tests that require direct access to the device will fail when run within a guest domain. However, running load-generation and other tests that do not require direct access to the device and/or driver within Linux and Windows guest domains is very valuable as such tests represent how the majority of load will be processed in actual XenServer installations.

When running tests in guest domains, ensure that you do so with the XenServer synthetic drivers installed in the guest domain. Installation of the synthetic drivers is a manual process for some guests. See XenServer Help for more details.

Tests that require an integrated build

One of the primary goals of the DDK is to allow partners to create, compile, and test their drivers with XenServer without requiring a "back-and-forth"of components with the XenServer engineering team.

However, some tests will only be possible after the driver RPMs and any accompanying binary RPMs have been supplied to Citrix and integrated into the XenServer product. Two examples are installing to, and booting from, Fibre Channel and iSCSI LUNs.

In these cases additional coordination is required after the components have been provided to Citrix to provide a pre-release XenServer build with the integrated components for testing.

Certification and support

Drivers

Citrix maintains a XenServer Hardware Compatibility List (HCL), found at https://hcl.xenserver.com. This lists all devices that have been tested and confirmed to function correctly with the XenServer product.

In order to be listed on the XenServer HCL, hardware vendors must utilize the appropriate certification kit, obtainable from https://hcl.xenserver.com/certkits/. The test kits contain a mix of manual and automated tests that are run on a host that contains the hardware to be certified. In general, two such hosts are required to perform the tests. Test results are submitted to Citrix for validation, and if they are approved, the device is listed on the HCL within a small number of working days, along with a link to the supplemental pack that contains any necessary driver, if this has not yet been incorporated into the XenServer product. In general, such supplemental packs will be hosted on partner web sites, though Citrix may additionally opt to link to (or host) the pack on its own Knowledge Base site.

For certification of converged or hybrid devices, such as CNAs, *each* function of the device must be separately certified. This implies that for a device with both networking and storage (HBA) functionality, both the networking certification tests and the storage certification tests must be carried out.

There is no restriction on who is permitted to submit certifications to the XenServer HCL, for example, it is *not* the case that only the hardware vendor can submit certifications for their products. Having said this, Citrix strongly prefers hardware vendors to perform certification testing, as they are best placed to test all of their products'features.

Once a device is listed on the HCL, Citrix will take support calls from customers who are using that device with XenServer. It is expected that partners who submit devices for inclusion in the HCL will collaborate with Citrix to provide a fix for any issue that is later found with XenServer which is caused by said device.

Userspace software

At present, supplemental packs that contain userspace software to be installed into dom0 may only be issued by partners who have agreements in place with Citrix where the partner provides level 1 and level 2 support to their customers.

The reason for this is because Citrix will not necessarily have had the opportunity to test a supplemental pack of a partner, and hence must rely on partner testing of the pack as installed on XenServer. Therefore, only partners who perform testing that has been agreed as sufficient by Citrix can ship supplemental packs. If a customer installs a pack that is not from an approved partner, their configuration will be deemed unsupported by Citrix: any issues found will need to be reproduced on a standard installation of XenServer, without the pack installed, if support is to be given.

Partners who want to produce supplemental packs that contain more than solely drivers must discuss this with their Citrix relationship manager as early as possible, in order to discuss what software is appropriate for inclusion, and what testing must be performed.

Additional Resources

May 16, 2023

In addition to supplemental packs, a variety of mechanisms are available for partners to interface with XenServer, and add value to the user experience. This chapter overviews the mechanisms, and provides web links to further information.

If pack authors have any questions concerning what or what not to include in a pack, or how a particular customization goal might be achieved, they are encouraged to contact their Citrix technical account manager.

Xen API plug-ins

Whilst the Xen API provides a wide variety of calls to interface with XenServer, partners have the opportunity to add to the API by means of XAPI plug-ins. These consist of Python scripts that are installed as part of supplemental packs, that can be run by using the `host.call_plugin` XAPI call. These plug-ins can perform arbitrary operations, including running commands in dom0, and making further XAPI calls, using the XAPI Python language bindings.

For examples of how XAPI plug-ins can be used, see the example plug-ins in the `/etc/xapi.d/plugins/` directory of a standard XenServer installation.

XenCenter plug-ins

XenCenter plug-ins provides the facility for partners to add new menus and tabs to the XenCenter administration GUI. In particular, new tabs can have an embedded web browser, meaning that existing web-based management interfaces can easily be displayed. When combined with Xen API plug-ins to drive new menu items, this feature can be used by partners to integrate features from their supplemental packs into one centralized management interface for XenServer.

To learn how to create plug-ins for XenCenter, see the samples and accompanying documentation in the XenCenter Plug-in Specification and Examples

repository. The XenCenter Plug-in Specification Guide
is available on the Developer Documentation site.

XenServer SDK

The Xen API is a Remote Procedure Call (RPC) based API providing programmatic
access to the extensive set of XenServer management features and tools.
Although it is possible to write applications that use the API directly through
raw RPC calls, the task of developing third-party applications is greatly
simplified by using language bindings exposing the individual API calls as
first-class functions in the target language. The XenServer SDK provides
language bindings for the C, C#, Java, Python, and PowerShell programming languages.

The XenServer SDK is shipped as a set of compiled libraries and source
code, which include a class for every API class and a method for each API call.
The libraries are accompanied by a number of test programs that can be used as
pedagogical examples. The XenServer SDK can be downloaded from
https://www.xenserver.com/downloads.

The XenServer Management API Reference
and the [XenServer Software Development Kit Guide](/en-us/xenserver/8/developer/sdk-guide.html
are available on the Developer Documentation site.

XenCenter Plug-in Specification Guide

December 9, 2023

This document explains how to write a plug-in for XenCenter, the GUI for
XenServer. Using the plug-in mechanism third-parties can:

- Create menu entries in the XenCenter menus linked to an executable file or
  PowerShell script, including full use of the XenServer PowerShell Module
  (XenServerPSModule) cmdlets.
- Cause a URL to be loaded into a tab in XenCenter.

The XenCenter plug-in mechanism is context aware, allowing you to use XenSearch
to specify complicated queries. Also, plug-ins can take advantage of contextual
information passed as arguments to executables or as replaceable parameters in
URLs.

A XenCenter plug-in consists of the following components:

- An XML configuration file.
- A resource DLL for each supported locale. Currently XenCenter exists in English and Japanese versions only.
- The application and any resources it requires.

Put these components of a plug-in in a subdirectory of the XenCenter installation directory. For example, a default installation of XenCenter requires that a plug-in reside in `C:\Program Files (x86)\XenServer\XenCenter\Plugins\<organization_name>\<plug-in_name>`

XenCenter loads all valid plug-ins found in subdirectories of the plug-ins directory when it starts:

- The plug-in name (<plug-in_name>) must be the same as the directory in which it is placed.
- The resource DLL and the XML configuration file must follow these naming conventions:
  - `<plug-in_name>.resources.dll`
  - `<plug-in_name>.xcplugin.xml`

For example, if your organization is called Citrix and you write a plug-in called Example which runs a batch file called `do_something.bat`, the following files must exist:

- `C:\Program Files (x86)\XenServer\XenCenter\Plugins\Citrix\Example\Example.resources.dll`
- `C:\Program Files (x86)\XenServer\XenCenter\Plugins\Citrix\Example\example.xcplugin.xml`
- `C:\Program Files (x86)\XenServer\XenCenter\Plugins\Citrix\Example\do_something.bat`

These paths assume that you use the default XenCenter installation directory.

XML Configuration File

May 10, 2023

Use your plug-in XML configuration file to define the menu items and tab items you want to appear in XenCenter. These items are referred to as *features*, and you can customize each one using the XML attributes described in this specification.

Example: A sample XML configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
   XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1"
      plugin_version="1.0.0.0">

  <MenuItem
        name="Hello World"
        menu="vm"
        contextmenu="none"
        serialized="obj"
        icon="Plugins\Citrix\HelloWorld\icon.png"
        search="dd7fbce2-b0d4-4c61-9707-e4b0f718673e"
        description="The world ' s friendliest plug-in, it loves to say
           hello">

    <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.exe" />

  </MenuItem>

  <TabPage
        name="Google"
        url="http://www.google.com/" />

  <Search
        uuid="dd7fbce2-b0d4-4c61-9707-e4b0f718673e"
        name="HelloSearch"
        major_version="2"
        minor_version="0"
        show_expanded="yes">

    <Query>

      <QueryScope>

        <VM />

      </QueryScope>

      <RecursiveXMOListPropertyQuery property="vm">

        <EnumPropertyQuery
              property="power_state"
              equals="no"
              query="Running" />

      </RecursiveXMOListPropertyQuery>

    </Query>

  </Search>

</XenCenterPlugin>
<!--NeedCopy-->
```

If your configuration file is invalid, XenCenter logs an error and the plug-in is ignored. You can enable, disable, and view errors with your plug-in configuration file in the **XenCenter plug-ins** dialog.

> **Note:**
>
> Errors are only shown in the dialog when your configuration file XML can be parsed. If your plug-in is not listed in the dialog, use the XenCenter log file to debug your XML.

Basic Structure

The XML elements in a configuration file have the following structure:

Figure *A hierarchy diagram. The top element is XenCenterPlugin. Its child elements are Search, MethodList, MenuItem, TabPage, GroupMenuItem. Search can contain XenSearch XML. MenuItem can have one of the following child elements: Shell, PowerShell, XenServerPowerShell. GroupMenuItem can contain multiple MenuItem elements.*

The `XenCenterPlugin` XML element is declared as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
    XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
    xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
    version="1">
...
</XenCenterPlugin>
<!--NeedCopy-->
```

With the following XML attributes:

> **Important:**
>
> The `version` attribute is required. If it is not set, your plug-in fails to load.

Key	Value	Description	Optional/Required	Default
`version`	1	The XenCenter plug-in version this plug-in was written for. Currently only version 1 is in use.	Required	-
`label`	[string]	Used in XenCenter as a user-friendly replacement for the plug-in name as dictated by the directory structure.	Optional	-
`description`	[string]	A short description of this plug-in to show in XenCenter.	Optional	-
`copyright`	[string]	A copyright notice for this plug-in to show in XenCenter	Optional	-
`link`	[string]	A URL web resource for the plug-in to show in XenCenter	Optional	-

XenSearch

You can include XenSearch definitions in your plug-in configuration file for features to reference. They can use these searches to tell XenCenter when they want to be enabled or shown.

Example: A MenuItem feature is using a XenSearch definition to restrict itself to shut down VMs:

```
<?xml version="1.0" encoding="UTF-8"?>

```

```
<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
    XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
    xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
    version="1">

  <MenuItem
      name="hello-menu-item"
      menu="vm"
      serialized="none"
      search="dd7fbce2-b0d4-4c61-9707-e4b0f718673e">
    <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.bat" />

  </MenuItem>

  <Search
      uuid="dd7fbce2-b0d4-4c61-9707-e4b0f718673e"
      name="NotRunning"
      major_version="2"
      minor_version="0"
      show_expanded="yes">

    <Query>

      <QueryScope>

        <VM />

      </QueryScope>

      <RecursiveXMOListPropertyQuery property="vm">

        <EnumPropertyQuery
            property="power_state"
            equals="no"
            query="Running" />

      </RecursiveXMOListPropertyQuery>

    </Query>

  </Search>

</XenCenterPlugin>
<!--NeedCopy-->
```

To get a XenSearch definition into your plug-in configuration file construct it in XenCenter, export it to a local file, and copy it into your configuration file.

Shared RBAC Method List

When the user is connected to a server running Role Based Access Control, your plug-in might not have permission to run all the server API calls you need. A method list can be defined for each command to warn XenCenter which API calls are needed before the plug-in is even run. See Commands and RBAC for more information.

By defining a shared RBAC method list as a child of your XenCenterPlugin node you can have multiple commands share a common list of methods:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
    XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1">

  <MenuItem
      name="hello-menu-item"
      menu="vm"
      serialized="none">

    <Shell
        filename="Plugins\Citrix\HelloWorld\HelloWorld.bat"
        required_method_list= ” methodList1 ”  />

  </MenuItem>

  <MenuItem
      name="hello-menu-item"
      menu="file"
      serialized="none">

    <Shell
        filename="Plugins\Citrix\HelloWorld\HelloWorld.bat"
        required_method_list= ” methodList1 ”  />

  </MenuItem>

  <MethodList name="methodList1">
      host.reboot, vm.start
  </MethodList>

</XenCenterPlugin>
<!--NeedCopy-->
```

This capability is purely for convenience and is equivalent to defining the method list on each command separately. When both `required_method_list` and `required_methods` are set on a command, `required_methods` takes precedence.

For syntax and further information see Commands and RBAC.

Features

May 10, 2023

Each XenCenter plug-in can define multiple features to extend the functionality of XenCenter for specific tasks. These features are the new menu items and tab pages you are adding to XenCenter.

Figure *The Feature element has child elements TabPageFeature, MenuItemFeature, and GroupMenuItemFeature.*

GroupMenuItem features are available to help organize your MenuItem features, but rely on MenuItem features to provide any functionality.

XML Attributes

All features share some common optional and required attributes that enable you to customize their appearance and functionality.

> **Important:**
>
> The `name` attribute is required for all features. If it is not set, your plug-in fails to load.

Key	Value	Description	Optional/Required	Default
`name`	[string]	The name for this feature. If the **`label`** attribute is not set, this name is used for display purposes in XenCenter. It is also used for logging.	Required	-

Key	Value	Description	Optional/Required	Default
`label`	[string]	Used as a `name` replacement for user facing display purposes in XenCenter.	Optional	-
`search`	[string]	The UUID of a XenSearch defined in your configuration file. It is used for setting enablement and visibility of this feature.	Optional	-
`description`	[string]	A short description of this feature.	Optional	-
`tooltip`	[string]	Any text you want to display a tooltip for this feature.	Optional	-
`icon`	[string]	A relative path from your XenCenter install directory to an icon image for this feature. It is displayed at size 16x16.	Optional	-

Example: A configuration file with a MenuItem feature using some of the common feature XML attributes

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
   XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1">

  <MenuItem
        name="Hello Exe World"
        menu="file"
        serialized="obj"
        icon="Plugins\Citrix\HelloWorld\icon.png"
        tooltip="Says hello to the whole world"
        label="Hello"
        search="dd7fbce2-b0d4-4c61-9707-e4b0f718673e"
        description="The world's friendliest plug-in, it loves to say
           hello">

    <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.exe"/>

  </MenuItem>

  <Search
        uuid="dd7fbce2-b0d4-4c61-9707-e4b0f718673e"
        name="HelloSearch"
        major_version="2"
        minor_version="0"
        show_expanded="yes">

    <Query>

      <QueryScope>

        <VM />

      </QueryScope>

      <RecursiveXMOListPropertyQuery property="vm">

        <EnumPropertyQuery
              property="power_state"
              equals="no"
              query="Running" />

      </RecursiveXMOListPropertyQuery>

    </Query>

  </Search>

</XenCenterPlugin>
<!--NeedCopy-->
```

MenuItem and GroupMenuItem

Plug-in authors can use MenuItem and GroupMenuItem features to add menu items in XenCenter. GroupMenuItems collect your menu items under sub menus and MenuItems launch your plug-in commands.

Figure: *An example hierarchy of menu items that launch various plug-in commands.*

- Each GroupMenuItem can have multiple MenuItem children
- Each MenuItem has exactly one child command which runs a target executable or script.

Example: A configuration file detailing the MenuItems and GroupMenuItems shown in the preceding figure

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
   XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1">

  <GroupMenuItem
        name="Hello World"
        menu="file">

    <MenuItem
          name="Hello PowerShell World"
          menu="file"
          serialized="obj">

      <PowerShell filename="Plugins\Citrix\HelloWorld\HelloWorld.ps1"
         />

    </MenuItem>

    <MenuItem
          name="Hello Batch World"
          menu="file"
          serialized="obj">

      <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.bat" />

    </MenuItem>

  </GroupMenuItem>

  <MenuItem
        name="Hello Exe World"
        menu="file"
        serialized="obj">

    <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.exe" />

  </MenuItem>

</XenCenterPlugin>
<!--NeedCopy-->
```

MenuItems which are children of a GroupMenuItem appear as a sub menu under their group. The MenuItem validation logic still requires that the 'menu'attribute is set for these sub MenuItems. However, the menu attribute on the parent GroupMenuItem dictates their location.

Figure *The File menu shows a menu item called Hello World with subitems called Hello PowerShell World and Hello Batch World.*

MenuItem XML Attributes

> **Important:**
>
> - The inherited feature attribute 'name'is required. If it is not set, your plug-in fails to load.
> - Each MenuItem feature must contain **exactly one** child node describing a XenCenter plug-in command, otherwise your plug-in does not load

Key	Value	Description	Optional/Required	Default
-	-	[All attributes inherited from feature]	-	-
`menu`	One of: file, view, pool, server, vm, storage, templates, tools, help	The XenCenter menu you would like this to appear under.	Required	-

Key	Value	Description	Optional/Required	Default
`serialized`	One of: obj, global	If set to obj, the menu item disables itself if its command is already running against the selected object. If set to global, only one instance of its command is allowed to run at a time regardless of what is selected.	Optional	-
`contextmenu`	One of: none, pool, server, vm, storage, template, folder	An extra context menu you would like this menu item to appear under. Unless you set 'none', the item is already present on the context menu that relates to the menu attribute (if such a context menu exists).	Optional	[value for menu]

GroupMenuItem XML Attributes

> **Important:**
>
> The inherited feature attribute 'name'is required. If it is not set, your plug-in fails to load.

Each GroupMenuItem feature can contain as many MenuItem child nodes as you would like.

Key	Value	Description	Optional/Required	Default
-	-	[All attributes inherited from feature]	-	-
`menu`	One of: File, view, pool, server, vm, storage, templates, tools, help	The XenCenter menu you would like this to appear under.	Required	-
`contextmenu`	One of: none, pool, server, vm, storage, template, folder	An extra context menu you would like this menu item to appear under. Unless you set 'none'the item is already present on the context menu that relates to the 'menu'attribute (if such a context menu exists).	Optional	[value for menu]

TabPage

Tab page features load a URL to display as an extra tab inside XenCenter. These tabs can be used to allow access to web management consoles or to add extra user interface features into XenCenter.

> **Note:**
>
> All local HTML and JavaScript examples in this section use the modified jQuery libraries for RPC calls through XenCenter in addition to the jQuery base library ? v1.3.2

Figure *A screenshot of the message board example plug-in.*

TabPage XML Attributes

> **Important:**
>
> - The inherited feature attribute 'name'is required. If it is not set, your plug-in fails to load.
> - The 'url'attribute is required. If it is not set, your plug-in fails to load.

Key	Value	Description	Optional/Required	Default
-	-	[All attributes inherited from feature]	-	-
`url`	[string]	The local or remote URL to load the HTML page from	Required	-
`context-menu`	true or false	Whether you would like the context menu for this HTML page to be enabled.	Optional	false
`xencenter-only`	true or false	If set, this TabPage appears when the XenCenter node is selected in the resource list and nowhere else.	Optional	false
`relative`	true or false	If set, the `url` attribute is interpreted as relative to the XenCenter install directory.	Optional	false
`help-link`	[string]	The URL to launch in a separate browser when the user requests for help on the tab page.	Optional	-

Key	Value	Description	Optional/Required	Default
`credentials`	true or false	Indicates that the webpage is using scripting and wants to use XenCenter's session credentials to interact with the server. This sets `window.external.SessionUuid` and `window.external.SessionUrl` for scripting access. **Warning:** By exposing these variables, you are allowing external webpages access to your server.	Optional	false
`console`	true or false	Indicates that this tab page is meant to replace the standard XenCenter console.	Optional	false

TabPage Javascript API

Using some modified jQuery libraries to pass XML-RPC calls through XenCenter it is possible for your tab page to communicate with the server using JavaScript. XenCenter provides a scripting object which contains the following public variables:

- SessionUuid
- SessionUrl

- SelectedObjectType
- SelectedObjectRef

These variables can be accessed through the `window.external` object in JavaScript and used to make server API calls:

```
// Retrieves the other config map for the currently selected XenCenter
object and passes it on to the callback function

function GetOtherConfig(Callback)

{

    var tmprpc;

    function GetCurrentOtherConfig()

    {

        var toExec = "tmprpc." + window.external.SelectedObjectType +
            ".get_other_config(Callback, window.external.SessionUuid
               window.external.SelectedObjectRef);";

        eval(toExec);

     }


    tmprpc= new $.rpc(
            "xml",
            GetCurrentOtherConfig,
            null,
            [window.external.SelectedObjectType + ".get_other_config"]

    );

 }

<!--NeedCopy-->
```

In this example we create an RPC object using 4 parameters:

1. Notifying that the RPC call is carrying XML (as opposed to JSON)
2. The function to run (`GetCurrentOtherConfig`) which fires off an API call; the modified jQuery library packages up the API call as an XML-RPC request and hands it to XenCenter.

   Notice that the function name for the callback is passed as an extra first parameter to the API call.
3. We don't specify a version number for the XML (**`null`**) which is interpreted

as 1.0.

4. We pass a description of the API call we want to make inside `GetCurrentOtherConfig` so the appropriate objects are created for the function to access.

Required Functions

> **Important:**
>
> It is required that you define a `RefreshPage` function in your JavaScript.

XenCenter calls this function every time it reloads the HTML page or adjusts the variables on the scripting object. Structure your code so that the `RefreshPage` function can easily tear down and rebuild the state of the page:

```
$(document).ready(RefreshPage);

function RefreshPage()

{

    // hide the error div and show the main content div

    $("#content").css({
 "display" : "" }
 );
    $("#errorContent").css({
 "display" : "none" }
 );
    $("#errorMessage").html("");
    RefreshMessagesAndStatus();
    RefreshDescription();
    RefreshTags();
 }

<!--NeedCopy-->
```

Setting the function to be called at `$(document).ready()` ensures there are no race conditions between XenCenter signaling to the page to refresh and the page itself being ready to receive these requests.

Receiving XenCenter Callbacks When you make an RPC object and get XenCenter to pass through an API call to the
server, you specify a callback function. When the XML-RPC request returns from the server, XenCenter invokes the callback function and passes in a JSON object that contains the result as a parameter. Look at `RefreshDescription` in the following example:

```
// The result object of any xmlrpc call to the server contains:
// - a result field which indicates whether it was succesfull or not,
// - a value field containing any returned data in json
// - an error description field containing any error information
// This function checks for success, displays any relevant errors, and
    returns
// a json object that corresponds to the value field

function CheckResult(Result)
{

    var myResult=Result.result;
    if(myResult.Status=="Failure")
    {

        var message=myResult.ErrorDescription\[0\];
        for(var i=1; i<myResult.ErrorDescription.length; i++)
        {

            message+=","+myResult.ErrorDescription\[i\];
         }

        $("#content").css({
 "display" : "none" }
 );
        $("#errorContent").css({
 "display" : "" }
 );
        $("#errorMessage").html(message);
        return;
     }

    if (myResult.Value == "")
    {

        return;
     }

    myResult = eval("("+myResult.Value+")");
    return myResult;
 }


// DESCRIPTION UPDATE SECTION
// This pair of methods chain to retrieve the description field from
    the server
// and display it. There is no writing to the description field on the
    server.
function RefreshDescription()
{

    var tmprpc;
    function RetrieveDescription()
    {

        var toExec = "tmprpc." + window.external.SelectedObjectType +
            ".get_name_description(ShowDescription, window.external.
               SessionUuid, window.external.SelectedObjectRef);";

        eval(toExec);
     }

    tmprpc= new $.rpc(
        "xml",
        RetrieveDescription,
        null,
        [window.external.SelectedObjectType + ".get_name_description"])

 }


function ShowDescription(DescriptionResult)
{

    var result = CheckResult(DescriptionResult);
    if (result == null)
    {

        $("#descriptionText").html("None");
        return;
     }

    $("#descriptionText").html(result);
 }

<!--NeedCopy-->
```

TabPage Replacement Consoles

This feature allows you to specify that your tab page feature replaces the standard console tab page in XenCenter. It is often used when a VM has its own web interface and the standard console tab page does not need to be seen.

To activate this feature, add the attribute `console="True"` to the `TabPageFeature` tag in your configuration file.

If XenCenter cannot reach the webpage you have specified in your tab page feature, the standard console tab page is returned and your tab page feature is hidden. If the tab page feature can be reached later on, it is automatically restored, and the standard console tab page hidden.

Commands

August 21, 2023

In your configuration file, each MenuItem feature has a single command as a child.
This child defines which executable or script to run when the user clicks the MenuItem.

This version of the XenCenter plug-in specification includes the following types of command:

- Shell
- PowerShell
- XenServerPowerShell

PowerShell and XenServerPowerShell are extensions of Shell and inherit all the properties of Shell. However, they both have extra features which make it easier to run PowerShell scripts. For example, XenServerPowerShell commands automatically load the XenServer PowerShell Module (XenServerPSModule) before running.

Figure *PowerShell and XenServerPowerShell extend Shell*

Parameter Sets

A parameter set is a collection of four parameters that describe which items are selected in the XenCenter resource list when your command is run.

While each command has its own way of receiving parameters from XenCenter, the parameters are always the same. They are delivered in sets of four that describe the selection in the XenCenter resource list: `url`, `sessionRef`, **`class`**, and `objUuid`.

Two of the parameters are used to allow communication to the relevant server:

- The `url` parameter indicates the address of the applicable standalone server or pool coordinator.
- The `sessionRef` parameter is the session opaque ref that can be used to communicate with this server.

Two of the parameters are used to describe which specific object is selected:

- The **`class`** parameter is used to show the class of the object which is selected in the resource list.
- The `objUuid` parameter is the UUID of this selected object.

Example: If you select both the local SR from a standalone server (Server A) and the pool node from a separate pool (Pool B), two parameter sets are passed into your plug-in:

Figure *An example of two sets of four parameters.*

In general, the plug-in receives one parameter set per object selected in the tree view, with two exceptions.

- Selecting a folder adds a parameter set per object in the folder, not the folder itself.
- Selected the XenCenter node adds a parameter set for each stand-alone server or pool that is connected, with the **`class`** and `objUuid` parameters marked with the keyword 'blank'.

  If the XenCenter node is selected, the plug-in receives the necessary information to perform actions on any connected servers. However, selecting this node provides no contextual information about what the user wants to target. The 'blank'keyword is used for the parameters that identify specifically what is selected.

Example: If you connect to Server A and Pool B from the previous example, and you selected both the XenCenter node and the local storage on Server A, you get the following parameter sets:

Figure *An example of multiple sets of parameters*

Shell

Shell commands are the most generic command type and launch executables, batch files, and other files which have a registered Windows extension.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
   XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1"
      plugin_version="1.0.0.0">

  <MenuItem
        name="hello-menu-item"
        menu="file"
        serialized="obj">

    <Shell
```

```
                  filename="Plugins\Citrix\HelloWorld\HelloWorld.bat"
                  window="true"
                  log_output="true"
                  dispose_time="0"
                  param="{
$type }
" />

  </MenuItem>

</XenCenterPlugin>
<!--NeedCopy-->
```

Shell Parameters

For Shell commands the parameter sets are passed through as command-line parameters to the batch file or executable. Any extra parameters supplied using the `param` XML attribute (see the following table) are first in the list of parameters, followed by sets of four command line parameters representing each parameter set.

Shell XML Attributes

> **Note:**
>
> The `filename` attribute is required and your plug-in only loads when it is set.

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`filename`	[string]	The file to execute, for example, an executable or a Windows batch file. The value is a relative path from your XenCenter installation directory.	Required	-	True

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`window`	true or false	Whether to start the process in a window or run it in the background.	Optional	true	-
`log_output`	true or false	Redirects the standard output and standard error streams of the plug-in process to the XenCenter log file.	Optional	false	-
`param`	[string]	A comma-separated string of extra parameters to pass to the process. You can pass a parameter with spaces by encasing it in XML-escaped quotes (&quot;)	Optional	-	True

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`dispose_time`	[float]	The grace period XenCenter gives the plug-in after it has requested it to cancel. After this period, XenCenter assumes the plug-in has hung and warns the user.	Optional	20.0	-
`required_methods`	[string]	A comma-separated string of API calls the plug-in wants to run. XenCenter blocks the plug-in launch and warns the user when these requirements are not met due to a role restriction under RBAC.	Optional	-	False

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`required_method_list`	[string]	The name of a list of methods called out in the MethodList node (child of XenCenterPlugin). If `required_methods` is set, this parameter is ignored.	Optional	-	False

For more information, see the section on RBAC protection.

PowerShell

PowerShell commands specifically target PowerShell scripts and have several enhancements over a basic Shell command to help you access the various parameters XenCenter can pass to your plug-in.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
    XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1"
      plugin_version="1.0.0.0">

  <MenuItem
        name="hello-menu-item"
        menu="file"
        serialized="obj" />

    <PowerShell
          filename="Plugins\Citrix\HelloWorld\HelloWorld.ps1"
          debug="true"
          window="true"
          log_output="true"
          dispose_time="0"
```

```
            param="{
 $type }
 "
            function="Write-Output {
 $type }
 ; read-host '[Press Enter to Exit]'" />

</MenuItem>

</XenCenterPlugin>
<!--NeedCopy-->
```

PowerShell Object Information Array

Information regarding the target items selected in the XenCenter resource list are stored in a PowerShell variable for easy access by your script. Inside the `$objInfoArray` variable is an array of hash maps, each representing a parameter set. Use the following keys to access the parameters in each hash map:

- `url`
- `sessionRef`
- **`class`**
- `objUuid`

```
[reflection.assembly]::loadwithpartialname('system.windows.forms')

foreach ($objInfo in $objInfoArray)
{

    $outputString = "url={
 0 }
 , sessionRef={
 1 }
 , objName={
 2 }
 , objUuid={
 3 }
 " `
      -f $objInfo["url"], $objInfo["uuid"], $objInfo["class"], $objInfo
          ["objUuid"]
    [system.Windows.Forms.MessageBox]::show("Hello from {
 0 }
 !" -f $outputString)
 }

<!--NeedCopy-->
```

PowerShell Extra Parameter Array

Any additional parameters you define using the `param` XML attribute inherited from Shell are stored in the `$ParamArray` variable as a simple array.

```
[reflection.assembly]::loadwithpartialname('system.windows.forms')

foreach ($param in $ParamArray)
{

    [system.Windows.Forms.MessageBox]::show("Hello from {
 0 }
 !" -f $param)
 }

<!--NeedCopy-->
```

PowerShell XML Attributes

> **Note:**
>
> The `filename` attribute inherited from Shell is required. Your plug-in only loads when this attribute points to a PowerShell script.

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
-	-	[All attributes inherited from Shell]	-	-	-
`debug`	true or false	Enables debugging output that traps and details any uncaught exceptions. It is highly recommended you set `window=true` if debug is enabled.	Optional	false	-

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`function`	[string]	Executes the provided string as PowerShell code after the main script has finished running.	Optional	-	True

XenServerPowerShell

In addition to the features provided by the PowerShell Command, the XenServerPowerShell Command loads the XenServer PowerShell Module (XenServerPSModule before running your target PowerShell script.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
   XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1"
      plugin_version="1.0.0.0">

  <MenuItem
        name="hello-menu-item"
        menu="file"
        serialized="obj" />

    <XenServerPowerShell
          filename="Plugins\Citrix\HelloWorld\HelloWorld.ps1"
          debug="true"
          window="true"
          log_output="true"
          dispose_time="0"
          param="{
 $type }
 "
          function="Write-Output {
 $type }
 ; read-host '[Press Enter to Exit]'" />

  </MenuItem>

</XenCenterPlugin>
```

```
<!--NeedCopy-->
```

XenServerPowerShell Initialization Details

The full setup done to prepare your PowerShell environment for communicating with the server is in the Initialize-Environment script in your XenServer PowerShell Module (XenServerPSModule) installation directory. When your target script begins running, the following setup happens:

- All the cmdlet aliases are initialized
- The global session variable is initialized to store your session information

XenServerPowerShell Parameters

The parameter sets and extra parameters can be accessed through the `$objInfoArray` and the `$ParamArray` variables as detailed in the previous PowerShell Command section.

XenServerPowerShell XML Attributes

> **Important:**
>
> The `filename` attribute inherited from Shell is required and your plug-in does only loads when it is set to point to a PowerShell script

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
-	-	[All attributes inherited from Shell]	-	-	-

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`debug`	true or false	Enables debugging output that traps and details any uncaught exceptions. It is highly recommended you set `window=true`	Optional	false	-
`function`	[string]	Executes the provided string as PowerShell code after the main script has finished running.	Optional	-	True

Preparing for Role Based Access Control (RBAC)

If you define the methods that a command requires in your configuration file, the command can be prepared for when XenCenter connects to a server that uses Role Based Access Control.

If the user does not have permission to run an API call on a server due to RBAC, the call fails with an `RBAC_PERMISSION_DENIED` exception. You can handle these exceptions from within the plug-in (examine the `ErrorDescription` field on the response for details). Alternatively, you can ask XenCenter to ensure that the user can run all possible commands you might need before the plug-in is run:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
   XENCENTERPLUGIN1//EN" "xencenter-1.dtd">

<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1"
      plugin_version="1.0.0.0">

  <MenuItem
        name="Hello Exe World"
        menu="file"
        serialized="obj"
        description="The worlds most friendly plug-in, it loves to say
            hello">

    <Shell
          filename="Plugins\Citrix\HelloWorld\HelloWorld.exe"
          required_methods="host.reboot, vm.start" />

  </MenuItem>

</XenCenterPlugin>
<!--NeedCopy-->
```

The `required_methods` attribute accepts a comma separated list of API calls in the format `object.method`.

If the user is operating on an RBAC enabled server, XenCenter checks that the user can run all of these API calls on their current role. If they can't, the plug-in is not launched and an error displayed:

Figure *An error in the Logs tab. The error is "Your current role is not authorized to perform this action on POOL NAME".*

Preparing for RBAC - Key White Lists

> **Important:**
>
> - Key white lists apply to advanced XenCenter keys. Only modify these lists if you know what you are doing.

In general, when operating under RBAC your role restricts you to modifying the `other-config` map on objects which are directly relevant to your role. For example, a VM Admin can modify the `other-config` map on a VM, but it cannot modify the `other-config` map on a server.

Some specific keys have been white listed to all roles above read-only. This setting allows XenCenter to set some advanced keys on `other-config` maps that are otherwise inaccessible to the user's role:

Target object	Key
VDI, SR, network, host, VM, pool	XenCenter.CustomFields.*
VDI, SR, network, host, VM, pool	folder
pool	EMPTY_FOLDERS
task	XenCenterUUID
task	applies_to
network	XenCenterCreateInProgress

You can enter these checks into your method list with the following
syntax:

```
    <MethodList name="methodList1">
        pool.set_other_config/key:folder,
        pool.set_other_config/key:XenCenter.CustomFields.*
    </MethodList>
<!--NeedCopy-->
```

Placeholders

If you use placeholders in your strings, XenCenter can call different functions,
use different URLs, or provide different parameters based on what object is
selected in the resource list.

When an XML attribute is marked as being able to accept placeholders you can leave
wildcards for XenCenter to fill in based on the properties of the object that is
selected in the resource list.

Placeholder	Description
`{ $type }`	The type of the selected object, for example, VM, Network
`{ $label }`	The label of the selected object
`{ $uuid }`	The UUID of the selected object, or the full pathname of a folder
`{ $description }`	The description of the selected object
`{ $tags }`	Comma-separated list of the tags of the selected object
`{ $host }`	The host name

Placeholder	Description
`{ $pool }`	The pool name
`{ $networks }`	Comma-separated list of the names of the networks attached to the object
`{ $storage }`	Comma-separated list of the names of the storage attached to the object
`{ $disks }`	Comma-separated list of the types of the storage attached to the object
`{ $memory }`	The host memory, in bytes
`{ $os_name }`	The name of the operating system that a VM is running
`{ $power_state }`	The VM power state, for example Halted, Running
`{ $virtualisation_status }`	The state of the pure virtualization drivers installed on a VM
`{ $start_time }`	Date and time that the VM was started
`{ $ha_restart_priority }`	The HA restart priority of the VM
`{ $size }`	The size in bytes of the attached disks
`{ $ip_address }`	Comma-separated list of IP addresses associated with the selected object
`{ $uptime }`	Uptime of the object, in a form such as '2 days, 1 hour, 26 minutes'
`{ $ha_enabled }`	true if HA is enabled, false otherwise
`{ $shared }`	Applicable to storage, true if storage is shared, false otherwise
`{ $vm }`	Comma-separated list of VM names
`{ $folder }`	The immediate parent folder of the selected object
`{ $folders }`	Comma-separated list of all the ancestor folders of the selected object

If the user has selected more than one target for the plug-in (by multiselect or by selecting a folder), it is not possible for XenCenter to know which object to use for the placeholder context. In a multi-target scenario all placeholders are substituted with the keyword `multi_target`. The plug-in can use this information to detect this situation.

Also, if there has been an error filling in a particular placeholder then the keyword **null** is substituted in to indicate the error. One example of where you see this keyword is if the XenCenter node was selected, which has no object properties to fill in.

Example: A community group adds their HTML help guides into a XenCenter tab based on which object is selected

```
<XenCenterPlugin
      xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
      version="1"
      plugin_version="1.0.0.0">

  <TabPage
        name="Extra Support"
        url="http://www.extra-help-for-xencenter.com/loadhelp.php?type
            ={
 $type }
 " />

</XenCenterPlugin>
<!--NeedCopy-->
```

Resources and Internationalization

May 9, 2023

In the following text, `<plug-in_name>` is the value of the plug-in name attributes and `<entry_name>` is the value of the MenuItem/TabPage name attributes.

XenCenter reads the following resources from `<plug-in>.resources.dll`:

- `<plug-in_name>.description` - Shown in the plug-ins dialog.
- `<plug-in_name>.copyright` - Vendor copyright statement. Shown in the plug-ins dialog.
- `<plug-in_name>.link` - Link to vendor's webpage. Shown in the plug-ins dialog.
- `<entry_name>.`**`label`** - The menu entry label.
- `<entry_name>.description` - Shown in the plug-ins dialog.
- `<entry_name>.icon` - The icon to use in the menu entry. This icon is a 16x16 PNG. Can be omitted.
- `<entry_name>.tooltip` - The tooltip to use when the menu entry is disabled. Can be omitted.

To create the resources file:

1. Create a RESX file containing the appropriate strings (You can do this task in any project in Visual Studio, for example, console project).
2. Open up Visual Studio command prompt and navigate to the RESX file's directory
3. Run `ResGen.exe <plug-in_name>.resx`

   (`ResGen.exe` is found in `C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin`).
4. Run

   `al.exe /t:lib /embed:<plug-in_name>.resources /out:<plug-in_name>.resources.dll`

   (`al.exe` is also found in `C:\Program Files\Microsoft SDKs\Windows\v7.0A\bin`).
5. If you want to compile extra resource DLLs for any specific cultures, edit the RESX file appropriately. Run these commands again with an extra `/culture` argument in the `al.exe` command specifying the two letter culture string. For example, for Japanese, run:

   `al.exe /t:lib /embed:<plug-in_name>.resources /culture:ja /out:<plug-in_name>.resources.dll`
6. Place the invariant culture `<plug-in_name>.resources.dll` into the `<plug-in_dir>` folder (with `<plug-in_name>.xcplugin.xml` and so on). Set up other cultures as follows: `<plug-in_dir>\<culture>\<plug-in_name>.resources.dll`
   where `<culture>` is the two-letter ISO culture name.

Unless stated otherwise, all of these entries are mandatory. If any are missing, then the problem is logged, and the menu option is disabled.

Deploying

December 9, 2023

Each plug-in is installed into: `<XenCenter_install_dir>\Plugins\<organization_name>\<plug-in_name>`.

> **Notes:**
>
> - The author's installation directory (`<XenCenter_install_dir>\Plugins\<`

> organization_name>)
> is known in this specification as <org_root>.
>
> - <org_root> is read-only at runtime.
> - By default <XenCenter_install_dir> is C:\Program Files (x86)\XenServer\XenCenter.
> - <organization_name> is the name of the organization or individual authoring the plug-in.
> - <plug-in_name> is the name of the plug-in.
> - <XenCenter_install_dir> can be looked up in the registry. In a default XenCenter installation, the XenCenter install directory can be found at HKEY_CURRENT_USER\SOFTWARE\XenServer\XenCenter\InstallDir. If XenCenter was installed for all users, this key is under HKEY_LOCAL_MACHINE.

In <org_root>\<plug-in_name>, ensure that the following items exist:

- A plug-in declaration file, named <plug-in_name>.xcplugin.xml.
- A satellite assembly for resources, named <plug-in_name>.resources.dll.
- Anything else that the plug-in needs for its proper functions.

Other than as specified in this document, XenCenter ignores the contents of <org_root>. Plug-in authors can install whatever content they require within their own <org_root>. You can, for example, have libraries or graphics that are shared between all plug-ins. In which case <org_root> (or a shared subdirectory of it) is a good place for these artifacts.

The same freedom applies to <org_root>\<plug-in_name>. Material in there that XenCenter does not recognize is ignored.

The <XenCenter_install_dir>\Plugins directory is scanned when XenCenter starts and plug-ins that are found are loaded. To rescan the directory, restart XenCenter or use the **Re-Scan Plug-in Directory** button on the **XenCenter plug-ins** dialog. Any subdirectory that does not contain <plug-in_name>.xcplugin.xml is silently ignored.

Data Governance

February 12, 2024

This article provides information regarding the collection, storage, and retention of logs by XenServer.

XenServer is a server virtualization platform that enables the customer to create and manage a deployment of virtual machines. XenCenter is the management UI for XenServer. XenServer and XenCenter can collect and store customer data as part of providing the following capabilities:

- **Telemetry** - The telemetry functionality transmits basic licensing information about a XenServer pool. XenServer collects this basic licensing data as necessary for its legitimate interests, including license compliance.
- **Server status reports** - A server status report can be generated on-demand and uploaded to Citrix Insight Services or provided to Support. The server status report contains information that can aid in diagnosing issues in the your environment.
- **Automatic updates for the Management Agent** - The Management Agent runs within VMs hosted on a XenServer host or pool. If the server or pool is licensed, the Management Agent can check for and apply updates to itself and to the I/O drivers in the VM. As part of checking for updates, the automatic update feature makes a web request to Cloud Software Group that can identify the VM where the Management Agent runs.
- **XenCenter check for updates** - This feature determines whether any hotfixes, cumulative updates, or new releases are available for the XenServer hosts and pools XenCenter manages. As part of checking for updates, this feature makes a web request to Citrix that includes telemetry. This telemetry is not user-specific and is used to estimate the total number of XenCenter instances worldwide.
- **XenCenter email alerts** XenCenter can be configured to send email notifications when alert thresholds are exceeded. To send these email alerts, XenCenter collects and stores the target email address.

Telemetry information received by Cloud Software Group is treated in accordance with our Agreements.

Telemetry

The XenServer telemetry functionality collects basic licensing information about your XenServer pools.

When you install XenServer, your pool coordinator gathers telemetry data and uploads it weekly to a Microsoft Azure Cloud environment located in the United States. This data does not identify individuals or customers and is sent securely over HTTPS on port 443 to `https://telemetry.ops.xenserver.com/`. No information other than the four elements identified below is collected or transmitted.

Access to this data is restricted to members of the XenServer Operations and Product Management teams.

Telemetry information received by Cloud Software Group is treated in accordance with our Agreements.

Telemetry collected

For each XenServer pool, the pool coordinator collects the following data:

Data collected	Description
UUID	A random unique ID for the telemetry data of this pool. This UUID is not the same as the pool UUID or any other existing identifier. It is not collected in server status reports.
Product version	The version of XenServer installed in this pool.
Sockets (per host)	The number of sockets this host has.
Edition (per host)	The type of license on this host.

This data does not identify individuals or customers and contains no personally identifiable information.

Viewing the telemetry data

The data that XenServer submits is logged on your pool coordinator in `/var/telemetry/telemetry.data`. This file is not collected in the server status logs.

Server status reports

During the course of operation a XenServer host collects and logs various information on the server where XenServer is installed. These logs can be collected as part of a server status report.

A server status report can be generated on-demand. You can upload these reports to Citrix Insight Services or provide them to Support. The server status report contains information that can aid in diagnosing issues in your environment.

Server status reports that are uploaded to Citrix Insight Services are stored in Amazon S3 environments located in the United States.

XenServer and XenCenter collect information from the following data sources:

- XenCenter

- XenServer hosts and pools
- Hosted VMs

You can select which data items are included in the server status reports. You can also delete any server status reports that are uploaded to your MyCitrix account on Citrix Insight Services.

Citrix Insight Services does not implement an automatic data retention for server status reports uploaded by the customer. The customer determines the data retention policy. You can choose to delete any server status reports that are uploaded to your MyCitrix account on Citrix Insight Services.

Data collected

A server status report can contain the following log files:

Log type	Contains PII?
`device-model`	yes
`fcoe`	yes
`firstboot`	yes
`network-status`	yes
`process-list`	yes
`xapi`	yes
`xenserver-databases`	yes
`control-slice`	maybe
`disk-info`	maybe
`hardware-info`	maybe
`high-availability`	maybe
`host-crashdump-logs`	maybe
`kernel-info`	maybe
`loopback-devices`	maybe
`message-switch`	maybe
`multipath`	maybe
`system-logs`	maybe
`v6d`	maybe
`xapi-clusterd`	maybe

Log type	Contains PII?
`xapi-debug`	maybe
`xcp-rrdd-plugins`	maybe
`xen-info`	maybe
`xenopsd`	maybe
`xenserver-config`	maybe
`xenserver-install`	maybe
`xenserver-logs`	maybe
`xha-liveset`	maybe
`yum`	if customized
`network-config`	if customized
`cron`	if customized
`blobs`	no
`block-scheduler`	no
`boot-loader`	no
`conntest`	no
`CVSM`	no
`pam`	no
`system-services`	no
`tapdisk-logs`	no
`VM-snapshot-schedule`	no
`xapi-subprocess`	no
`xen-bugtool`	no
`xenserver-domains`	no

Management Agent automatic updates

The Management Agent runs within VMs hosted on a XenServer host or pool. If the host or pool is licensed, the Management Agent can check for and apply updates to itself and to the I/O drivers in the VM. As part of checking for updates, the automatic update feature makes a web request to us that can identify the VM where the Management Agent runs.

The web logs captured from the requests made by the Management Agent automatic updates feature are located in a Microsoft Azure Cloud environment located in the United States. These logs are then copied to a log management server in the United Kingdom.

The web requests made by the Management Agent automatic updates feature are made over HTTPS. Web log files are transmitted securely to the log management server.

You can select whether your VM uses the Management Agent automatic update feature. If you choose to use the Management Agent automatic update feature, you can also choose whether the web request includes the VM identifying information.

Web logs containing information from web requests made by the Management Agent automatic updates feature and the XenCenter check for updates feature can be retained indefinitely.

Data collected

The Management Agent automatic updates web requests can contain the following data points:

Data collected	Description	What we use it for
IP address	The IP address of the VM where the Management Agent is installed	
Partial VM UUID	The first four characters of the unique user ID for the VM where the Management Agent is installed	

XenCenter check for updates

This feature determines whether any hotfixes, cumulative updates, or new releases are available for the XenServer hosts and pools XenCenter manages. As part of checking for updates, this feature makes a web request to Cloud Software Group that includes telemetry. This telemetry does not personally identify users and is used to estimate the total number of XenCenter instances worldwide.

The web logs captured from the requests made by the XenCenter check for updates feature are located in a Microsoft Azure Cloud environment located in the United States. These logs are then copied to a log management server in the United Kingdom.

The web requests made by the XenCenter check for updates feature are made over HTTPS. Web log files are transmitted securely to the log management server.

The XenCenter check for updates feature is enabled by default. You can choose to disable this feature.

Data collected

The check for updates feature web requests contain the following data points:

Data collected	Description	What we use it for
IP address	The IP address of the XenCenter host machine	
XenCenter version	The version of XenCenter making the request	

XenCenter email alerts

XenCenter can be configured to send email notifications when alert thresholds are exceeded. To send these email alerts, XenCenter collects and stores the target email address.

The email address that XenCenter uses to send email alerts is stored on the machine where you installed XenCenter.

You can delete email alerts configured in XenCenter to remove the stored email information.

XenCenter retains the email information used to provide email alerts for the lifetime of the email notification. When you delete the configured email alert, the data is removed.

Data collected

To provide email alerts XenCenter stores the following data points:

Data collected	Description	What we use it for
Email address	The email address for alerts	To send alert and notification emails to
SMTP server	The SMTP server to use	To route the email alerts to the recipient