XenServer

Develop for XenServer

Contents

XenServer Software Development Kit Guide

March 5, 2024

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.

Overview of the XenServer Management API

May 30, 2024

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

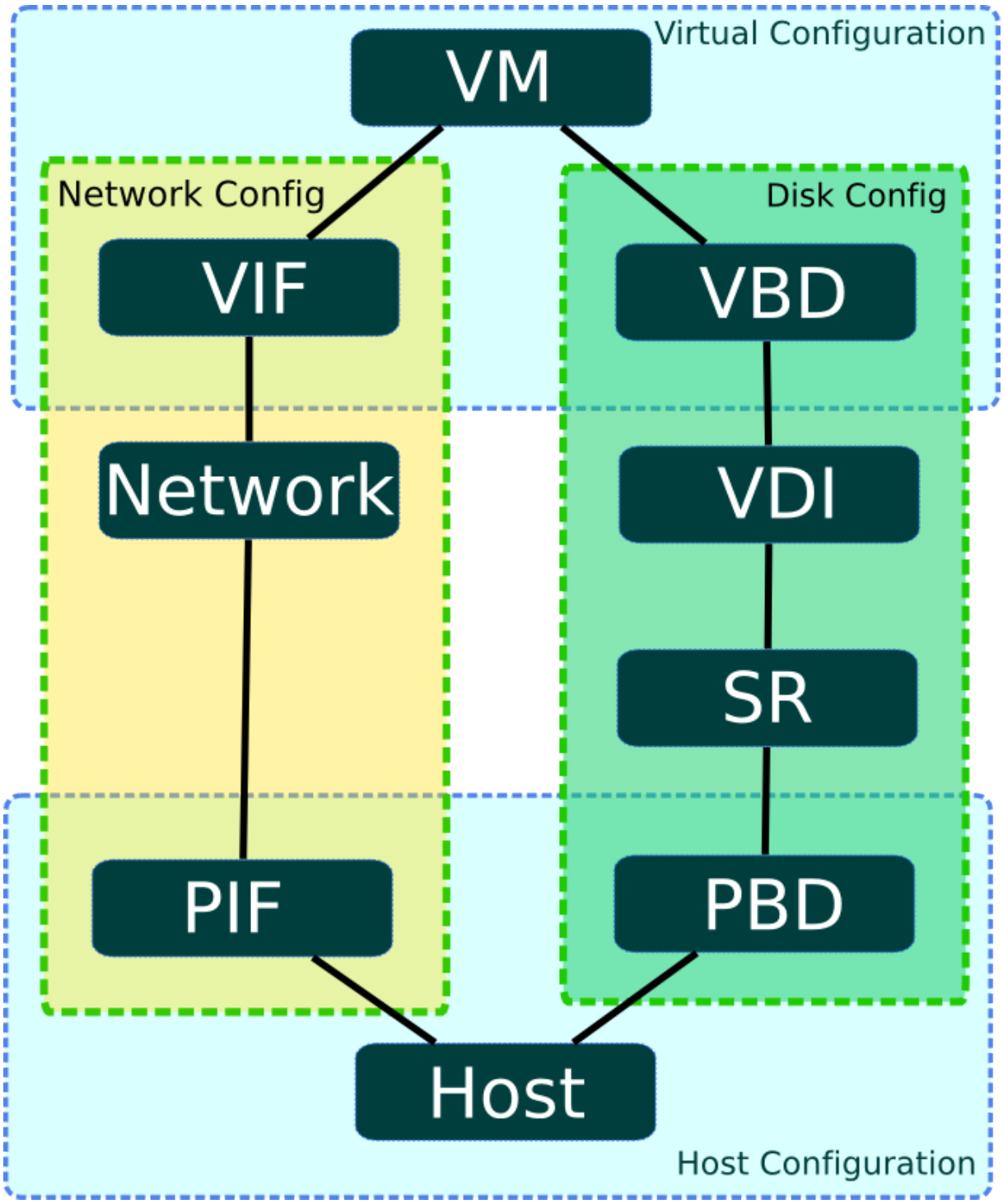
PIFs abstract both physical interfaces and VLANs (the latter distinguished by the existence of a positive integer in the "VLAN"field).

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.

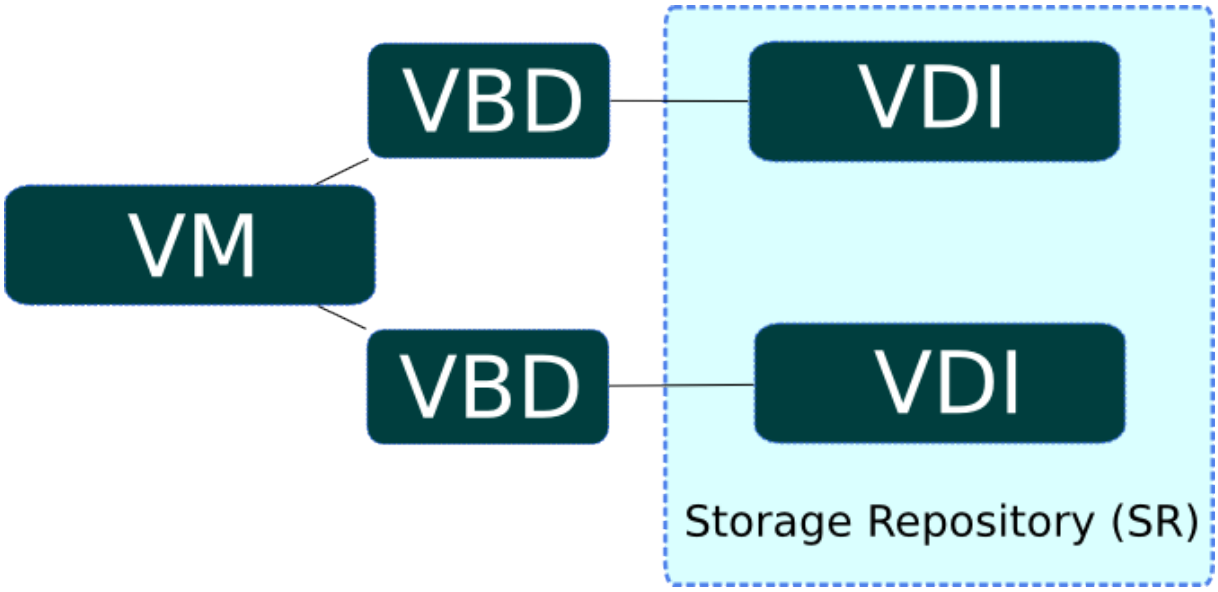
Attaching the disk image to a VM So far we have a running VM (that we assumed the existence of at the start of this example) and a fresh VDI that we just created. Right now, these are both independent objects that exist on the XenServer Host, but there is nothing linking them together. So our next step 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.

Getting Started

May 30, 2024

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 contents

The extracted contents of the SDK ZIP file are in the `XenServer-SDK` directory. 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.

Supported languages

The XenServer SDK provides support for the following languages:

- Powershell
- Python
- Java
- C #
- C

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.

Getting Started with PowerShell

May 28, 2024

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.

Using the XenServer PowerShell module

May 30, 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:

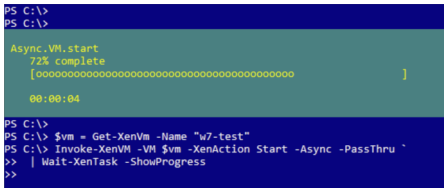
```
PS> $gpuGroups = Get-XenGPUGroup
PS> Get-XenVGPUtype | % {
 Get-XenGPUGroupProperty $gpuGroups[0] -XenProperty RemainingCapacity -
    VgpuType $_ }


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-->
```


Getting Started with Python

May 30, 2024

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.

Getting Started with Java

April 18, 2024

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.

Getting Started with C

April 18, 2024

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.

Getting Started with C

April 18, 2024

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.

Using the API

May 28, 2024

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

May 30, 2024

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.

Parameter	Description
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).
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
>    0000000000000000000
> -rwx------+ 1 user1 Domain Users      40 Aug  4 09:50
>    0000000000000000000.checksum
> -rwx------+ 1 user1 Domain Users 1048576 Aug  4 09:50
>    0000000000000000001
> -rwx------+ 1 user1 Domain Users      40 Aug  4 09:50
> ```



```
      0000000000000000001.checksum
-rwx------+ 1 user1 Domain Users 1048576 Aug  4 09:50
      0000000000000000003
-rwx------+ 1 user1 Domain Users      40 Aug  4 09:50
      0000000000000000003.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:

```
1 wget https://<server>/host_rrd?session_id=<session_opaque_reference>
2 <!--NeedCopy-->
```

To download a VM RRD, you must also specify the VM's `uuid` as a query parameter:

```
1 wget https://<server>/vm_rrd?session_id=<session_opaque_reference>&uuid
     =<vm_uuid>
2 <!--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.

XML tag	Description
`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 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-->
```

XenServer Management API extensions

May 30, 2024

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
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:

```
1 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.

Other extensions

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.other_config["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.
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.
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.

Key	Semantics
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.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.

Management API reference

April 18, 2024

This section provides the API reference information for all supported versions of XenServer and Citrix Hypervisor:

- XenServer 8
- Citrix Hypervisor 8.2 Cumulative Update 1

API overview

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 errorcode) 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

XenServer 8 Management API

April 18, 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

April 18, 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:

```
1  (VM ref set)  VM.get_all()
2  <!--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:

```
1      <value><string>destroy</string></value>
2  <!--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

```
1      (task ref set)  Task.get_all(session ref session_id)
2  <!--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:

```
1  $ python2.7
2  >>>
3  <!--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

May 28, 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

May 28, 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:

```
1 blob ref create (session ref session_id, string mime_type, bool public)
2 <!--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_coefficien	**`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:

```
1 float get_token_timeout (session ref session_id, Cluster ref self)
2 <!--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:

```
1 float get_token_timeout_coefficient (session ref session_id, Cluster
     ref self)
2 <!--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:

```
1 void disable (session ref session_id, Cluster_host ref self)
2 <!--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:

```
1 void enable (session ref session_id, Cluster_host ref self)
2 <!--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:

```
1 void add_to_other_config (session ref session_id, crashdump ref self,
     string key, string value)
2 <!--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 -batch-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:

```
1 void add_to_other_config (session ref session_id, host_cpu ref self,
     string key, string value)
2 <!--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:

```
1 string enable_thin_provisioning (session ref session_id, host ref host,
      SR ref SR, int initial_allocation, int allocation_quantum)
2 <!--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:

```
1 LVHD ref get_by_uuid (session ref session_id, string uuid)
2 <!--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:

```
1 LVHD record get_record (session ref session_id, LVHD ref self)
2 <!--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:

```
1 void add_to_guest_agent_config (session ref session_id, pool ref self,
     string key, string value)
2 <!--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:

```
1 void add_to_gui_config (session ref session_id, pool ref self, string
     key, string value)
2 <!--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:

```
1 void crl_install (session ref session_id, string name, string cert)
2 <!--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:

```
1 void enable_tls_verification (session ref session_id)
2 <!--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:

```
1 pool ref set get_all (session ref session_id)
2 <!--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:

```
1 (pool ref -> pool record) map get_all_records (session ref session_id)
2 <!--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-custom-uefi -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:

```
1 string get_uuid (session ref session_id, pool_update ref self)
2 <!--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:

```
1 VDI ref get_vdi (session ref session_id, pool_update ref self)
2 <!--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:

```
1 PVS_cache_storage ref create (session ref session_id, PVS_cache_storage
      record args)
2 <!--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:

```
1 blob ref create_new_blob (session ref session_id, SR ref sr, string
     name, string mime_type, bool public)
2 <!--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:

```
1 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)
2 <!--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:

```
1 string probe (session ref session_id, host ref host, (string -> string)
      map device_config, string type, (string -> string) map sm_config)
2 <!--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:

```
1 user ref get_by_uuid (session ref session_id, string uuid)
2 <!--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:

```
1 string get_fullname (session ref session_id, user ref self)
2 <!--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_implementation	*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:

```
1 VGPU_type ref set get_all (session ref session_id)
2 <!--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:

```
1 string set get_qos_supported_algorithms (session ref session_id, VIF
     ref self)
2 <!--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:

```
1 VIF record get_record (session ref session_id, VIF ref self)
2 <!--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:

```
1 void add_tags (session ref session_id, VM ref self, string value)
2 <!--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_VERSIO, 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:

```
1 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)
2 <!--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_ALLO
, 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:

```
1 void set_VCPUs_params (session ref session_id, VM ref self, (string ->
     string) map value)
2 <!--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:

```
1 void set_xenstore_data (session ref session_id, VM ref self, (string ->
     string) map value)
2 <!--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_S

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:

```
1 string get_uuid (session ref session_id, VM_appliance ref self)
2 <!--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

April 18, 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-->
```

Citrix Hypervisor 8.2 Management API

April 18, 2024

This document defines the Citrix Hypervisor 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, and so on. Such additional RPCs are described explicitly in the API reference.

Wire Protocol for Remote API Calls

April 18, 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:

```
1 (VM ref set)  VM.get_all()
2 <!--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 can't make assumptions about the concrete form of these strings and can't 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 can't make assumptions about the concrete form of these strings and can't 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"
```



```
 9     }
10
11 <!--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`, and so on.) 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:

```
 1   {
 2
 3     "result": [
 4         "OpaqueRef:604f51e7-630f-4412-83fa-b11c6cf008ab",
 5         "OpaqueRef:670d08f5-cbeb-4336-8420-ccd56390a65f"
 6     ],
 7     "error": null,
 8     "id": "xyz"
 9    }
10
11 <!--NeedCopy-->
```

while the return value of the same call made on a logged out session may look like this:

```
 1   {
 2
 3     "result": null,
 4     "error": [
 5         "SESSION_INVALID",
 6         "OpaqueRef:93f1a23cd-a640-41e3-b163-10f86e0eae67"
 7     ],
```



```
    "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`, and so on.) 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 can't make any assumptions or attempt to parse the string contents. References are not guaranteed to be permanent identifiers for objects; clients can't 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 user name 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 user name 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). 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 user name 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). 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

May 28, 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` must 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

May 28, 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

Name	Description
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
PBD	The physical block devices through which hosts access SRs
PCI	A PCI device
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

Name	Description
PVS_site	machines serving blocks of data for provisioning VMs
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
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

Name	Description
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

object.field	*object.field*	*relationship*
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
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

object.field	*object.field*	*relationship*
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
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.

enum after_apply_guidance	
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 cls	
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

enum cluster_host_operation	
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
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
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	
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
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 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

enum ipv6_configuration_mode	
None	Do not acquire an IPv6 address
Static	Static IPv6 address configuration

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 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, consult the underlying VLAN slave PIF.

enum pool_allowed_operations	
cluster_create	Indicates this pool is in the process of creating a cluster
designate_new_master	Indicates this pool is in the process of changing master
ha_disable	Indicates this pool is in the process of disabling HA
ha_enable	Indicates this pool is in the process of enabling HA

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
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, for example, rebuilding mirrors.

enum sriov_configuration_mode	
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
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

enum storage_operations	
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
pending	task is in progress
success	task was completed successfully

enum tristate_type	
no	Known to be false
unspecified	Unknown or unspecified
yes	Known to be true

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 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

enum vdi_type	
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

enum vif_locking_mode	
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”
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.

enum vm_operations	
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”
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”
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”

enum vm_operations	
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

enum vmpp_archive_target_type	
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 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 user name 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 (for example, username, organization) 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
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
`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 fingerprint / hash
host	`host ref`	*RO/constructor*	The host where the certificate is installed
not_after	`datetime`	*RO/constructor*	Date before which the certificate is valid
not_before	`datetime`	*RO/constructor*	Date after which the certificate is valid
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

type	name	description
`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_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_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_coefficien	**`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:

```
1 void disable (session ref session_id, Cluster_host ref self)
2 <!--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:

```
1 void enable (session ref session_id, Cluster_host ref self)
2 <!--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:

```
1 void add_to_other_config (session ref session_id, crashdump ref self,
     string key, string value)
2 <!--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
iscsi_iqn	string	*RO/constructor*	The initiator IQN for the host
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
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
other_config	(string -> string)map	*RW*	additional configuration

Field	Type	Qualifier	Description
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
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
software_version	(string -> string)map	*RO/constructor*	version strings

Field	Type	Qualifier	Description
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
uefi_certificates	string	*RO/constructor*	The UEFI certificates allowing Secure Boot
updates	pool_update ref set	*RO/runtime*	Set of updates
updates_requiring_reboo	pool_update ref set	*RO/runtime*	List of updates which require reboot
uuid	string	*RO/runtime*	Unique identifier/object reference

Field	Type	Qualifier	Description
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: 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
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: pool-operator

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_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_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: pool-operator

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)
<!--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: 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_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_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_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_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_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_uefi_certificates *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

type	name	description
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:

```
1 string set list_methods (session ref session_id)
2 <!--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:

```
1 void local_management_reconfigure (session ref session_id, string
     interface)
2 <!--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: 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: 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_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_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_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 *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
stepping	string	*RO/runtime*	**Deprecated**. the stepping of the physical CPU
utilisation	**float**	*RO/runtime*	**Deprecated**. the current CPU utilisation

Field	Type	Qualifier	Description
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
`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
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:

```
1 void add_to_other_config (session ref session_id, host_crashdump ref
     self, string key, string value)
2 <!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
host_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 *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

Field	Type	Qualifier	Description
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
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
`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: 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
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

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
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

type	name	description
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

Field	Type	Qualifier	Description
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: 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 wich 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
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
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.
default_SR	SR ref	*RW*	Default SR for VDIs
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

Field	Type	Qualifier	Description
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
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

Field	Type	Qualifier	Description
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
live_patching_disabled	bool	*RW*	The pool-wide flag to show if the live patching feauture is disabled or not.
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
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

Field	Type	Qualifier	Description
redo_log_vdi	`VDI ref`	*RO/runtime*	indicates the VDI to use for the redo-log other than when HA is enabled
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
uefi_certificates	`string`	*RW*	The UEFI certificates allowing Secure Boot
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
wlb_verify_cert	`bool`	*RW*	true if communication with the WLB server should enforce TLS certificate verification.

RPCs associated with class: pool

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

type	name	description

Minimum Role: pool-operator

Return Type: **void**

RPC name: certificate_install *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

Minimum Role: pool-operator

Return Type: **void**

RPC name: certificate_list *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 *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: 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 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 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 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_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: pool-operator

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_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_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
`pool ref`	pool	The pool whose external authentication should be enabled
`(string -> string)map`	config	A list of key-values containing the configuration data

type	name	description
`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: pool-operator

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: 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_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_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_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_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_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_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_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_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_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 *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 user name used to authenticate with the wlb server
string	wlb_password	The password used to authenticate with the wlb server
string	xenserver_username	The user name used by the wlb server to authenticate with the xenserver

type	name	description
string	xenserver_password	The password used by the wlb server to authenticate with the xenserver

Minimum Role: pool-operator

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 host name of the master of the pool to join
string	master_username	The user name 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 host name of the master of the pool to join
string	master_username	The user name 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_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: 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_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_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_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_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_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_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_uefi_certificates *Overview:*

Set the uefi_certificates field of the given pool.

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	reference to the object
`string`	value	New value to set

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: pool-operator

Return Type: **`void`**

RPC name: set_wlb_verify_cert *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: 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

type	name	description
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

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

Field	Type	Qualifier	Description
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
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

type	name	description
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:

Run 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
`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

Field	Type	Qualifier	Description
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:*

Run 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:

```
1 PVS_cache_storage ref create (session ref session_id, PVS_cache_storage
      record args)
2 <!--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 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
name_description	string	*RW*	a notes field containing human-readable description

Field	Type	Qualifier	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
`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: role

A set of permissions associated with a subject

Fields for class: role

Field	Type	Qualifier	Description
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_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.

Field	Type	Qualifier	Description
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.
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
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.

Field	Type	Qualifier	Description
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

type	name	description
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_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

Field	Type	Qualifier	Description
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
`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:

```
1 blob ref create_new_blob (session ref session_id, SR ref sr, string
     name, string mime_type, bool public)
2 <!--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:

```
1 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)
2 <!--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:

```
1 string probe (session ref session_id, host ref host, (string -> string)
      map device_config, string type, (string -> string) map sm_config)
2 <!--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_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_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
status	(string -> string)map	*RW*	Status information about the tunnel
transport_PIF	PIF ref	*RO/constructor*	The interface used by the tunnel

Field	Type	Qualifier	Description
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)
<!--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

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_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_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
uuid	`string`	*RO/runtime*	Unique identifier/object reference

Field	Type	Qualifier	Description
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/runtime*	is the device currently attached (erased on reboot)
device	string	*RO/runtime*	device seen by the guest, for example, 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, for example, 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

type	name	description
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: 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:

```
1 void remove_from_other_config (session ref session_id, VDI ref self,
     string key)
2 <!--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:

```
1 void remove_from_sm_config (session ref session_id, VDI ref self,
     string key)
2 <!--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_implementation	*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:

```
1 VGPU_type ref set get_all (session ref session_id)
2 <!--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/runtime*	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:

```
1 string set get_qos_supported_algorithms (session ref session_id, VIF
     ref self)
2 <!--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:

```
1 VIF record get_record (session ref session_id, VIF ref self)
2 <!--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

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*	**Deprecated**. 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/runtime*	describes the CPU flags on which the VM was last booted
last_booted_record	`string`	*RO/runtime*	marshalled value containing VM record at time of last boot, updated dynamically to reflect the runtime state of the domain
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).

Field	Type	Qualifier	Description
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.
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

Field	Type	Qualifier	Description
NVRAM	(string -> string)map	*RO/constructor*	initial value for guest NVRAM (containing UEFI variables, and so on). 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
PCI_bus	string	*RW*	**Deprecated**. PCI bus path for pass-through devices
platform	(string -> string)map	*RW*	platform-specific configuration
power_state	vm_power_state	*RO/runtime*	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

Field	Type	Qualifier	Description
recommendations	string	*RW*	An XML specification of recommended values and ranges for properties of this VM
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, for example, 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)

Field	Type	Qualifier	Description
snapshot_info	(string -> string)map	*RO/runtime*	Human-readable information concerning this snapshot
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/runtime*	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

Field	Type	Qualifier	Description
user_version	**int**	*RW*	Creators of VMs and templates may store version information here.
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, for example, 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_VERSIO, 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:

```
1 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)
2 <!--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:

```
1 void forget_data_source_archives (session ref session_id, VM ref self,
     string data_source)
2 <!--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:

```
1 on_crash_behaviour get_actions_after_crash (session ref session_id, VM
     ref self)
2 <!--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_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, for example, 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 deprecated.**

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_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 running 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 running 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_ALLO
, 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 user name 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

Minimum Role: vm-power-admin

Return Type: **void**

Possible Error Codes: VM_BAD_POWER_STATE, OTHER_OPERATION_IN_PROGRESS, VM_IS_TEMPLATE, OPERATION_NOT_ALLOWED, VM_MIGRATE_FAILED

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
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

type	name	description
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: 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.
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: vm-power-admin

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_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
(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

type	name	description
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.
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

type	name	description
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 *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: vm-operator

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)
<!--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

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_S

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

type	name	description
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-power-admin

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: vm-operator

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.

Field	Type	Qualifier	Description
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, for example, blocked, dying.
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 run

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 run

Minimum Role: pool-operator

Return Type: `string`

An XMLRPC result

Class: VTPM

A virtual TPM device

Fields for class: VTPM

Field	Type	Qualifier	Description
backend	VM ref	*RO/constructor*	the domain where the backend is located
uuid	string	*RO/runtime*	Unique identifier/object reference
VM	VM ref	*RO/constructor*	the virtual machine

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, VTPM record args)
<!--NeedCopy-->
```

Arguments:

type	name	description
session ref	session_id	Reference to a valid session
VTPM record	args	All constructor arguments

Minimum Role: vm-admin

Return Type: VTPM ref

reference to the newly created object

RPC name: destroy *Overview:*

Destroy the specified VTPM instance.

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	reference to the object

Minimum Role: vm-admin

Return Type: **void**

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_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.
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
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

April 18, 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;
16  please wait a while and try again. If the problem persists, please
      contact your
17  support representative.<h1> Additional information </h1>Jsonrpc.
      Malformed_metho
18 d_request(&amp;quot;{
19  jsonrpc=...,method=...,id=... }
20  &amp;quot;)</body></html>
21 <!--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:

```
1      <struct>
2        <member>
3          <name>Status</name>
4          <value>Failure</value>
5        </member>
6        <member>
7          <name>ErrorDescription</name>
8          <value>
9            <array>
10             <data>
11               <value>MAP_DUPLICATE_KEY</value>
12               <value>Customer</value>
13               <value>eSpiel Inc.</value>
14               <value>eSpiel Incorporated</value>
15             </data>
16           </array>
17         </value>
18       </member>
19     </struct>
20 <!--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:

```
1  {
2
3      "jsonrpc": "2.0",
4      "error": {
5
```



```
          "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-->
```

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

This properties of this PIF cannot be changed. Only the properties of non-bonded physical PIFs, or bond masters 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 master.

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 master 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-->
```

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-->
```

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-->
```

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 master 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 is fenced (it is not in the master'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 master 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 master.

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 pool member. Please pass API calls via the master host.

Signature:

```
HOST_IS_SLAVE(Master IP address)
<!--NeedCopy-->
```

HOST_ITS_OWN_SLAVE

The host is its own pool member. Please use pool-emergency-transition-to-master or pool-emergency-reset-master.

No parameters.

HOST_MASTER_CANNOT_TALK_BACK

The master reports that it cannot talk back to the pool member 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 master says the server is not known to it. Is the server in the master's database and pointing to the correct master? Are all servers using the same pool secret?

Signature:

```
HOST_UNKNOWN_TO_MASTER(host)
<!--NeedCopy-->
```

ILLEGAL_VBD_DEVICE

The specified VBD device is not recognized: 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_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_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_UPDATE

The uploaded update package is invalid.

Signature:

```
INVALID_UPDATE(info)
<!--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 master 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.

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.

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_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 master.

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 master.

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 master.

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.

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_INVALID

You gave an invalid session reference. It may have been invalidated by a server restart, or timed out. 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 pool member cannot be disabled because the pool member 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-->
```

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-->
```

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-->
```

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_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-->
```

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 sub-strate

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-->
```


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, 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-->
```

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 Changed Block Tracking Guide

March 5, 2024

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

May 28, 2024

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

March 5, 2024

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 28, 2024

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:

```
1 is_cbt_enabled = session.xenapi.VDI.get_cbt_enabled(<vdi_ref>)
2 <!--NeedCopy-->
```

xe command line:

```
1 xe vdi-param-get param-name=cbt-enabled uuid=<vdi-uuid>
2 <!--NeedCopy-->
```

Deleting VDI snapshot data and retaining the snapshot metadata

May 28, 2024

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 28, 2024

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 28, 2024

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 28, 2024

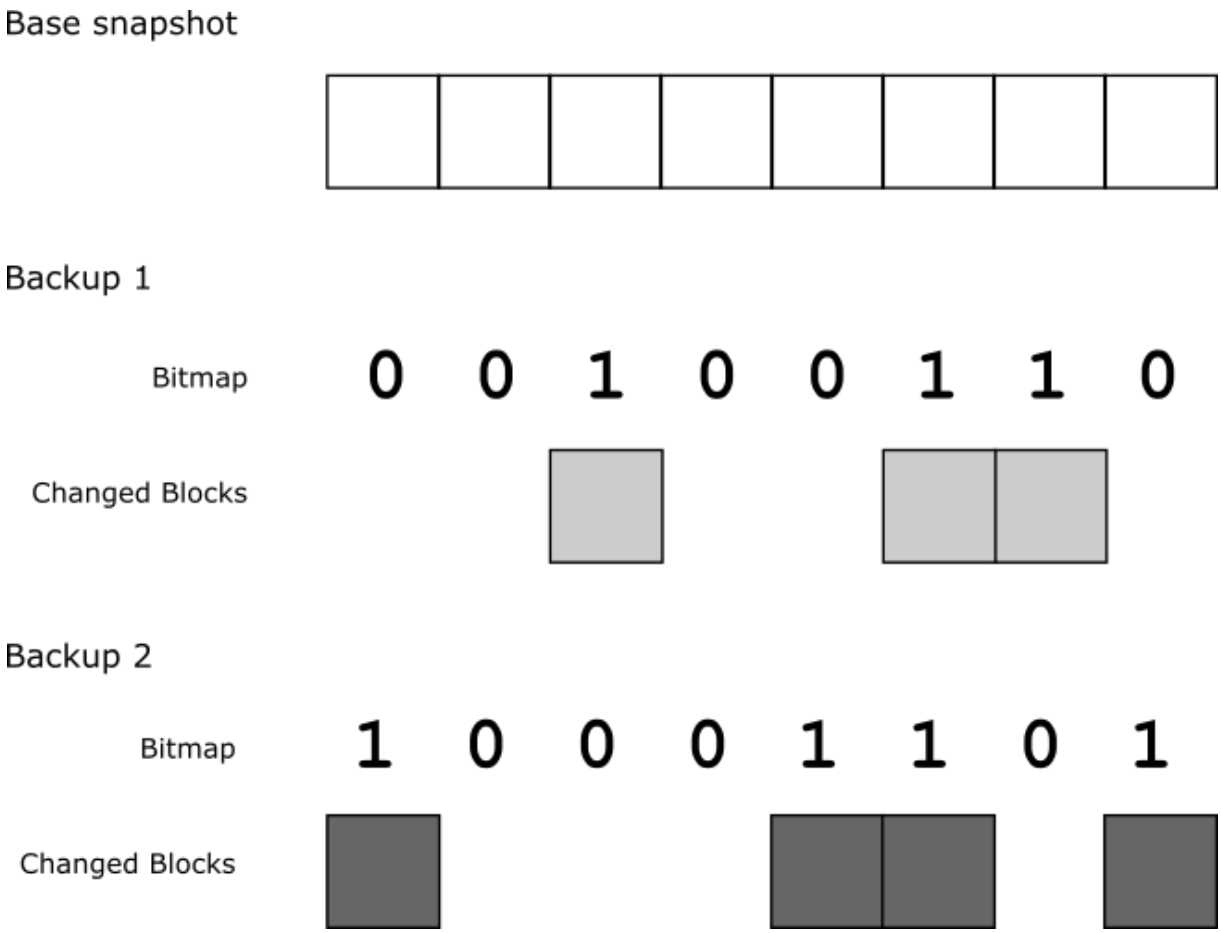
When using backed up data to restore the state of a VDI, you must import a full VDI into XenServer. You cannot import only the sets of changed blocks.
To import incremental backups created with changed block tracking data into XenServer, you must first use these incremental backups to create a full VDI.

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:

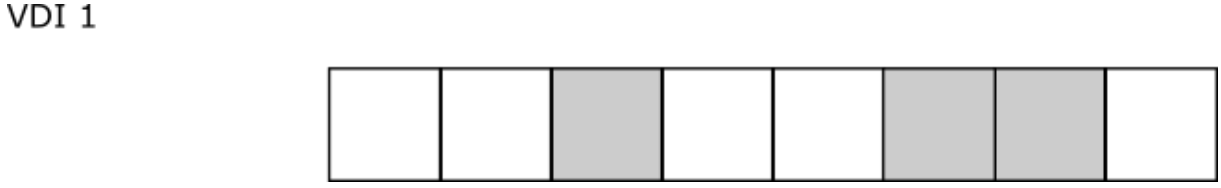
- A base snapshot that captures the data for the full VDI.
- Backup 1: The first incremental backup, which consists of a bitmap list of blocks changed since the base snapshot and the data for only those changed blocks.
- Backup 2: The second incremental backup, which consists of a bitmap list of blocks 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".

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:

```
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 28, 2024

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

May 28, 2024

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

March 5, 2024

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

March 5, 2024

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

May 28, 2024

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

March 5, 2024

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

March 5, 2024

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:

```
1  --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:

```
1 mkdir packages
2 cd packages
3 cp /root/rpmbuild/RPMS/x86_64/helloworld-1.0-1.x86_64.rpm .
```

The pack metadata is then created, using the script.

```
1 build-update --uuid 3fbce6cf-5cd2-4d32-9602-e8122c562169 --label
     example pack --version 1.0.0 \
2 --description "An Example Pack" --base-requires "product version=8.0.0"
      \
```



```
3 --key "Example Updates (update) <example@example.com>" --keyfile /root/
     RPM-GPG-KEY-XS-DDK-TEST \
4 -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

March 5, 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

March 5, 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 http://www.citrix.com/ready/hcl. 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 28, 2024

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/developer/sdk-guide.html
are available on the Developer Documentation site.

XenCenter Plug-in Specification Guide

March 5, 2024

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 28, 2024

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:

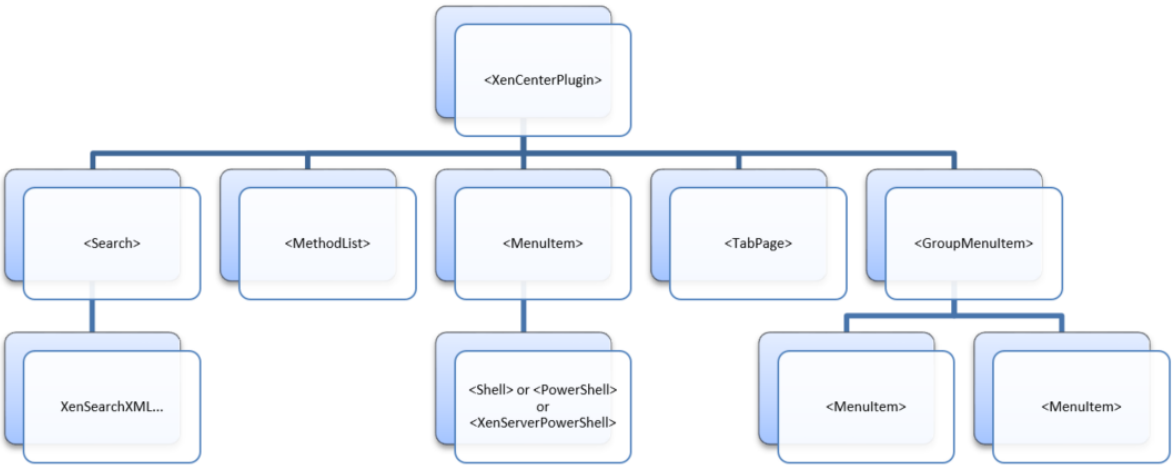


Figure *A hierarchy diagram. The top element is XenCenterPlugin. Its child elements are Search, MethodList, MenuItem, TabPage, GroupMenuItem. Search can contain XenSearch XML. MenuItem can have one of the following child elements: Shell, PowerShell, XenServerPowerShell. GroupMenuItem can contain multiple 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
```



```
6       xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
7       version="1">
8   ...
9   </XenCenterPlugin>
10  <!--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>
```



```
43
44    </Search>
45
46  </XenCenterPlugin>
47  <!--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:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!DOCTYPE XenCenterPlugin PUBLIC "-//XENCENTERPLUGIN//DTD
       XENCENTERPLUGIN1//EN" "xencenter-1.dtd">
4
5  <XenCenterPlugin
6        xmlns="http://www.citrix.com/XenCenter/Plugins/schema"
7        version="1">
8
9    <MenuItem
10       name="hello-menu-item"
11       menu="vm"
12       serialized="none">
13
14     <Shell
15         filename="Plugins\Citrix\HelloWorld\HelloWorld.bat"
16         required_method_list=”methodList1” />
17
18   </MenuItem>
19
20   <MenuItem
21       name="hello-menu-item"
22       menu="file"
23       serialized="none">
24
25     <Shell
26         filename="Plugins\Citrix\HelloWorld\HelloWorld.bat"
27         required_method_list=”methodList1” />
28
29   </MenuItem>
```



```
30
31    <MethodList name="methodList1">
32        host.reboot, vm.start
33    </MethodList>
34
35  </XenCenterPlugin>
36  <!--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 28, 2024

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.

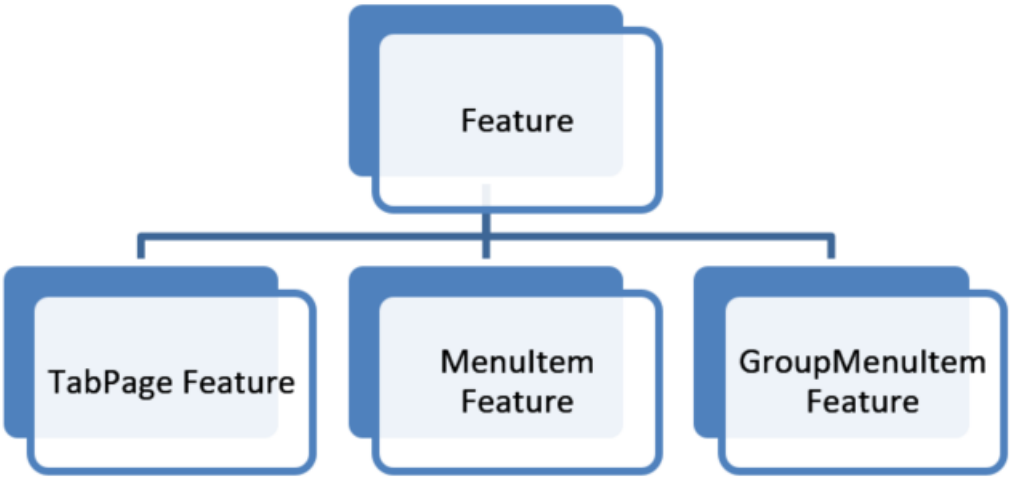


Figure *The Feature element has child elements TabPageFeature, MenuItemFeature, and GroupMenuItemFeature.*

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	-
`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	-

Key	Value	Description	Optional/Required	Default
`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">
```



```
29
30      <Query>
31
32        <QueryScope>
33
34          <VM />
35
36        </QueryScope>
37
38        <RecursiveXMOListPropertyQuery property="vm">
39
40          <EnumPropertyQuery
41                property="power_state"
42                equals="no"
43                query="Running" />
44
45        </RecursiveXMOListPropertyQuery>
46
47      </Query>
48
49    </Search>
50
51  </XenCenterPlugin>
52  <!--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.

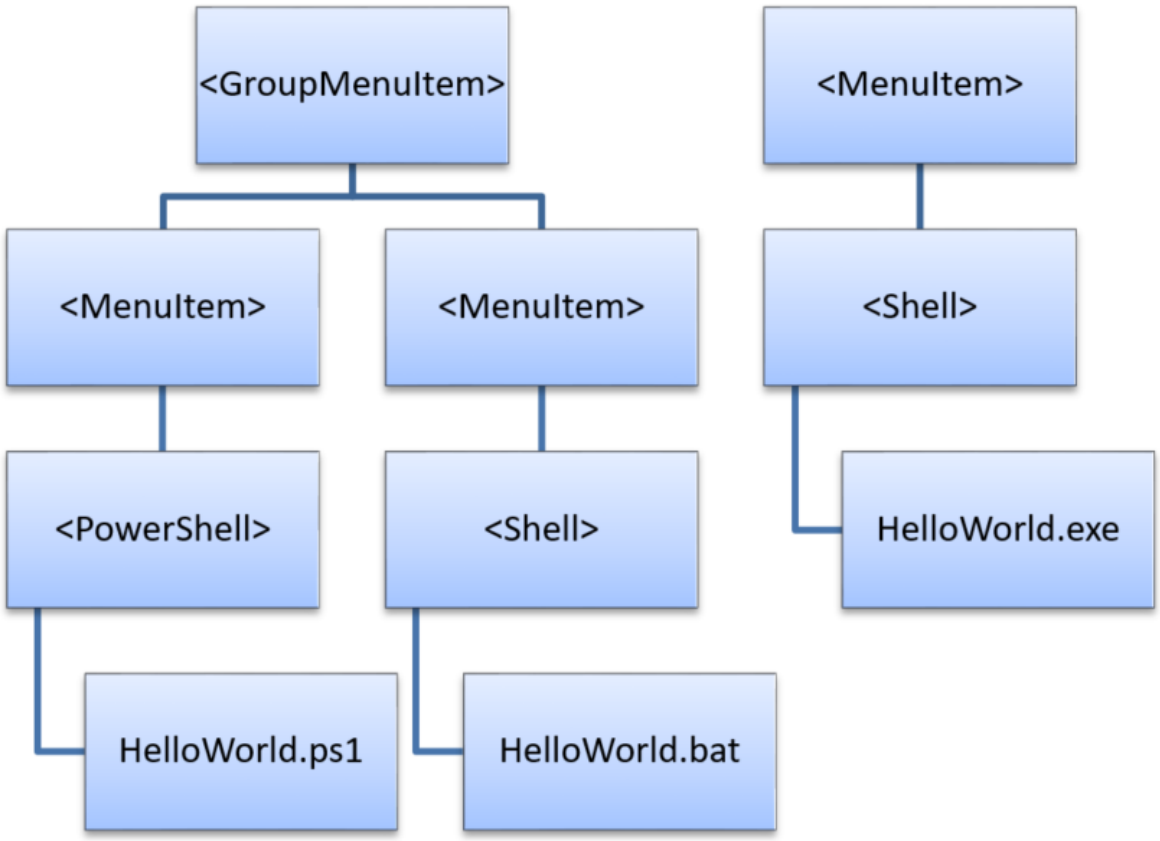


Figure: *An example hierarchy of menu items that launch various plug-in commands.*

- 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"
```



```
16            serialized="obj">
17
18        <PowerShell filename="Plugins\Citrix\HelloWorld\HelloWorld.ps1"
              />
19
20      </MenuItem>
21
22      <MenuItem
23            name="Hello Batch World"
24            menu="file"
25            serialized="obj">
26
27        <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.bat" />
28
29      </MenuItem>
30
31    </GroupMenuItem>
32
33    <MenuItem
34          name="Hello Exe World"
35          menu="file"
36          serialized="obj">
37
38      <Shell filename="Plugins\Citrix\HelloWorld\HelloWorld.exe" />
39
40    </MenuItem>
41
42  </XenCenterPlugin>
43  <!--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.

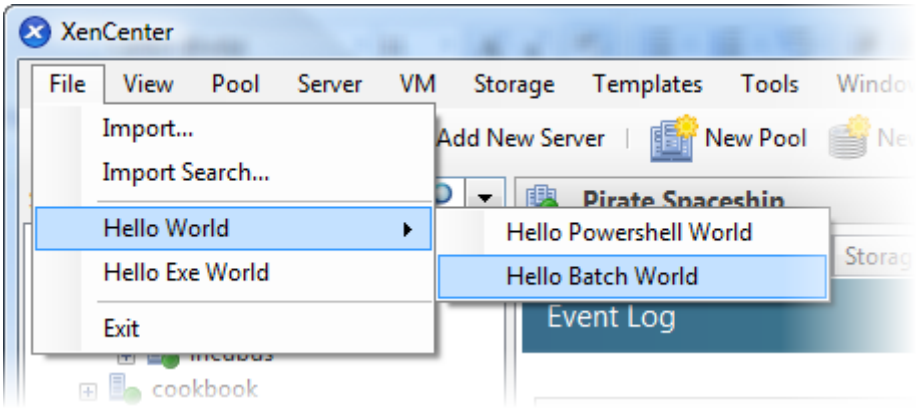


Figure *The File menu shows a menu item called Hello World with subitems called Hello PowerShell World and Hello Batch World.*

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	-
`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	-

Key	Value	Description	Optional/Required	Default
`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	-

Key	Value	Description	Optional/Required	Default
`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

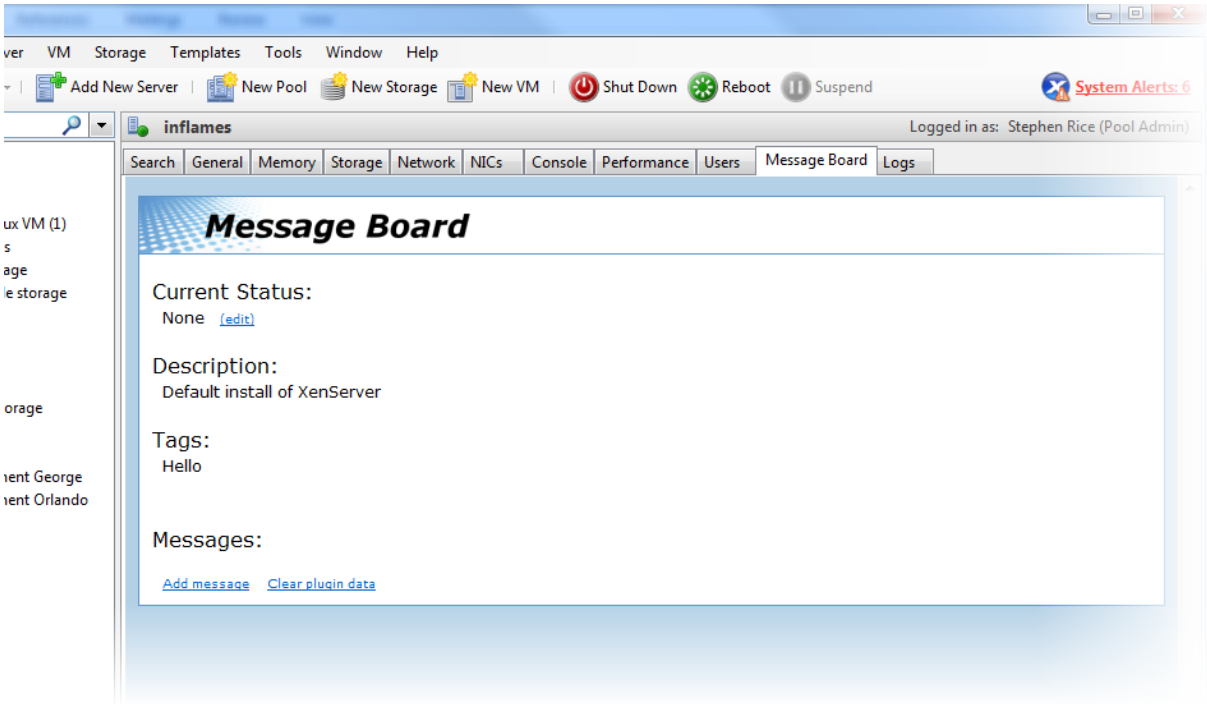


Figure *A screenshot of the message board example plug-in.*

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	-

Key	Value	Description	Optional/Required	Default
`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()
```



```
51      {
52
53          var toExec = "tmprpc." + window.external.SelectedObjectType +
54              ".get_name_description(ShowDescription, window.external.
                    SessionUuid, window.external.SelectedObjectRef);";
55
56          eval(toExec);
57       }
58
59      tmprpc= new $.rpc(
60          "xml",
61          RetrieveDescription,
62          null,
63          [window.external.SelectedObjectType + ".get_name_description"])
64
65   }
66
67
68  function ShowDescription(DescriptionResult)
69  {
70
71      var result = CheckResult(DescriptionResult);
72      if (result == null)
73      {
74
75          $("#descriptionText").html("None");
76          return;
77       }
78
79      $("#descriptionText").html(result);
80   }
81
82  <!--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

May 28, 2024

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.

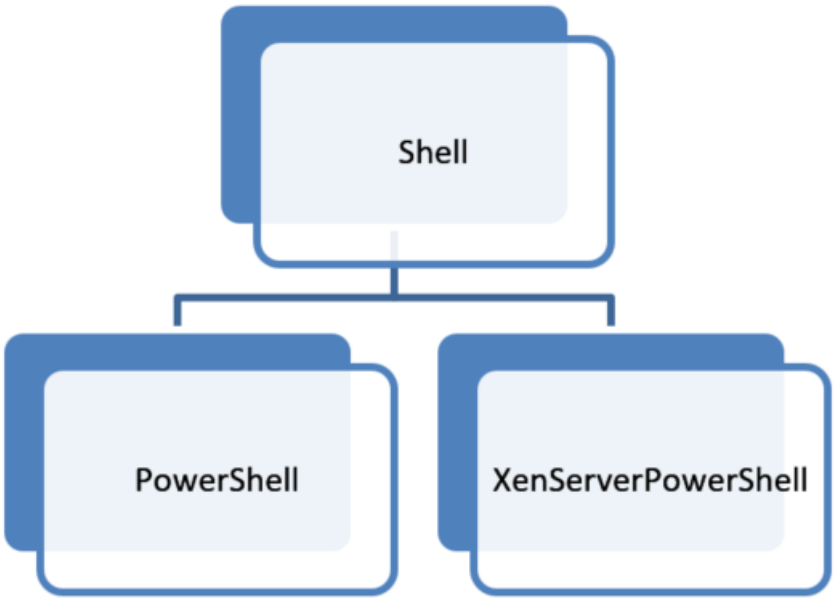


Figure *PowerShell and XenServerPowerShell extend Shell*

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 }
```



```
21   " />
22
23    </MenuItem>
24
25 </XenCenterPlugin>
26 <!--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
`window`	true or false	Whether to start the process in a window or run it in the background.	Optional	true	-

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`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 (")	Optional	-	True
`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	-

Key	Value	Description	Optional/Required	Default	Accepts Placeholders
`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
`required_methodlist`	[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	-
`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">
```



```
 8
 9    <MenuItem
10          name="hello-menu-item"
11          menu="file"
12          serialized="obj" />
13
14      <XenServerPowerShell
15            filename="Plugins\Citrix\HelloWorld\HelloWorld.ps1"
16            debug="true"
17            window="true"
18            log_output="true"
19            dispose_time="0"
20            param="{
21   $type }
22   "
23            function="Write-Output {
24   $type }
25   ; read-host '[Press Enter to Exit]'" />
26
27    </MenuItem>
28
29  </XenCenterPlugin>
30  <!--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]	-	-	-
`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:

```
1 <?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:

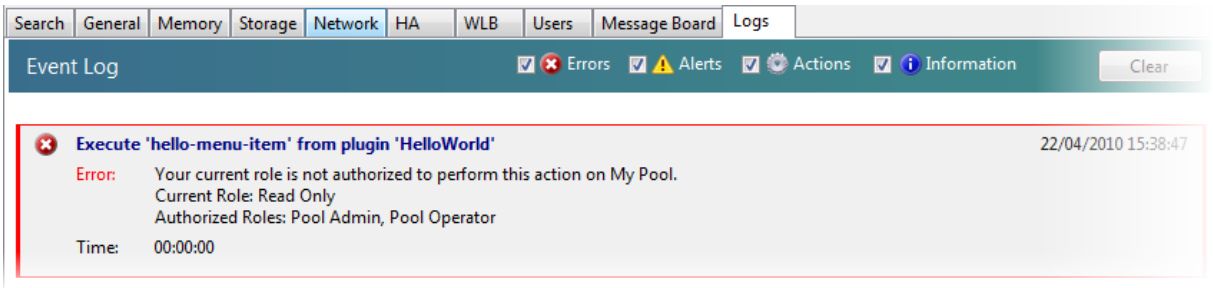


Figure *An error in the Logs tab. The error is "Your current role is not authorized to perform this action on POOL NAME".*

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
`{ $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

Placeholder	Description
`{ $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

March 5, 2024

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

March 5, 2024

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.

