Xen Management API

Version: API Revision 0.9.0
Date: 5th April 2007
Candidate for Release
Comments are welcome!

Ewan Mellor: ewan@xensource.com
Richard Sharp: richard.sharp@xensource.com
David Scott: david.scott@xensource.com

Contributors:

Stefan Berger, IBM
Daniel Berrangé, Red Hat
Gareth Bestor, IBM
Hollis Blanchard, IBM
Mike Day, IBM
Jim Fehlig, Novell
Jon Harrop, XenSource
Vincent Hanquez, XenSource
John Levon, Sun Microsystems
Jon Ludlam, XenSource
Alastair Tse, XenSource
Daniel Veillard, Red Hat
Tom Wilkie, University of Cambridge

Copyright © 2006-2007 XenSource, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Chapter 1

Introduction

This document contains a description of the Xen 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, with a wire format based upon XML-RPC. No specific language bindings are prescribed, although examples will be 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_{run}: the field is Read Only. Furthermore, its value is automatically computed at runtime. For example: current CPU load and disk IO throughput.
- RO_{ins}: 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.

A full list of types is given in Chapter 2. However, there are three types that require explicit mention:

- $t\ Ref$: signifies a reference to an object of type t.
- $t\ Set$: signifies a set containing values of type t.
- $(t_1, t_2)\ Map$: signifies a mapping from values of type t_1 to values of type t_2.

Note that there are a number of cases where *Ref*s are *doubly linked*—e.g. 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 explicitly lists 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 existance 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).

1.1 RPCs associated with fields

Each field, `f`, has an RPC accessor associated with it that returns `f`'s value:

- "`get_f(Ref x)`": takes a `Ref` that refers to an object and returns the value of `f`.

Each field, `f`, with attribute *RW* and whose outermost type is *Set* has the following additional RPCs associated with it:

- an "`add_to_f(Ref x, v)`" RPC adds a new element v to the set[1];
- a "`remove_from_f(Ref x, v)`" RPC removes element `v` from the set;

Each field, `f`, with attribute *RW* and whose outermost type is *Map* has the following additional RPCs associated with it:

- an "`add_to_f(Ref x, k, v)`" RPC adds new pair `(k, v)` to the mapping stored in `f` in object `x`. Adding a new pair for duplicate key, `k`, overwrites any previous mapping for `k`.
- a "`remove_from_f(Ref x, k)`" RPC removes the pair with key `k` from the mapping stored in `f` in object `x`.

Each field whose outermost type is neither *Set* nor *Map*, but whose attribute is *RW* has an RPC acessor associated with it that sets its value:

- For *RW* (*R*ead/*W*rite), a "`set_f(Ref x, v)`" RPC function is also provided. This sets field `f` on object `x` to value `v`.

1.2 RPCs associated with classes

- Each class has a *constructor* RPC named "`create`" that takes as parameters all fields marked *RW* and RO_{ins}. 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)`" RPC that returns a set of objects of that class that have the specified `label`.
- Each class has a "`destroy(Ref x)`" RPC that explicitly deletes the persistent object specified by `x` 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.

1.2.1 Additional RPCs

As well as 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.

[1] Since sets cannot contain duplicate values this operation has no action in the case that `v` was already in the set.

1.3 Wire Protocol for Remote API Calls

API calls are sent over a network to a Xen-enabled host using the XML-RPC protocol. In this Section we describe how the higher-level types used in our API Reference are mapped to primitive XML-RPC types.
In our API Reference we specify the signatures of API functions in the following style:

```
(ref_vm Set)   VM.get_all()
```

This specifies that the function with name `VM.get_all` takes no parameters and returns a Set of `ref_vm`s. These types are mapped onto XML-RPC types in a straight-forward manner:

- Floats, Bools, DateTimes and Strings 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`, etc).
- ints are all assumed to be 64-bit in our API and are 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, a value of `destroy` of type `on_normal_exit`, would be conveyed as:

```
<value><string>destroy</string></value>
```

- for all our types, `t`, our type `t Set` simply maps to XML-RPC's `Array` type, so for example a value of type `cpu_feature 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>
```

- 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, double) Map` containing a 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>
```

- our `Void` type is transmitted as an empty string.

1.3.1 Note on References vs UUIDs

References are opaque types — encoded as XML-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, u, and returns an opaque reference to the server-side object that has UUID=u;
- A `get_uuid` function (a regular "field getter" RPC) that takes an opaque reference, r, and returns the UUID of the server-side object that is referenced by r.

1.3.2 Return Values/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 `Status` was set to `Success` then the Struct contains a second element named `Value`:

- The element of the struct named `Value` contains the function's return value.

In the case where `Status` is set to `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 remainder of the array are strings representing error parameters relating to that code.

For example, an XML-RPC return value from the `host.get_resident_VMs` function above 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>
```

1.4 Making XML-RPC Calls

1.4.1 Transport Layer

The following transport layers are currently supported:

- HTTP/S for remote administration
- HTTP over Unix domain sockets for local administration

1.4.2 Session Layer

The XML-RPC interface is session-based; before you can make arbitrary RPC calls you must login and initiate a session. For example:

```
session_id    session.login_with_password(string uname, string pwd)
```

Where `uname` and `password` refer to your username and password respectively, as defined by the Xen administrator. The `session_id` returned by `session.login_with_password` is passed to subequent RPC calls as an authentication token.
A session can be terminated with the `session.logout` function:

```
void          session.logout(session_id session)
```

1.4.3 Synchronous and Asynchronous invocation

Each method call (apart from methods on "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 in Section 1.3.2.
Only synchronous API calls are listed explicitly in this document. All asynchronous versions are in the special `Async` namespace. For example, synchronous call `VM.clone(...)` (described in Chapter 2) has an asynchronous counterpart, `Async.VM.clone(...)`, that is non-blocking.
Instead of returning its result directly, an asynchronous RPC call returns a `task-id`; this identifier is subsequently used to track the status of a running asynchronous RPC. Note that an asychronous call may fail immediately, before a `task-id` has even been created—to represent this eventuality, the returned `task-id` is wrapped in an XML-RPC struct with a `Status`, `ErrorDescription` and `Value` fields, exactly as specified in Section 1.3.2.
The `task-id` is provided in the `Value` field if `Status` is set to `Success`.
The RPC call

```
(ref_task Set)   Task.get_all(session_id s)
```

returns a set of all task IDs known to the system. The status (including any returned result and error codes) of these tasks 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.

1.5 Example interactive session

This section describes how an interactive session might look, using the python XML-RPC client library.
First, initialise python and import the library `xmlrpclib`:

```
\$ python2.4
...
>>> import xmlrpclib
```

Create a python object referencing the remote server:

```
>>> xen = xmlrpclib.Server("http://test:4464")
```

Acquire a session token by logging in with a username and password (error-handling ommitted for brevity; the session token is pointed to by the key `'Value'` in the returned dictionary)

```
>>> session = session.login_with_password("user", "passwd")['Value']
```

When serialised, this call looks like the following:

```
<?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>
  </params>
</methodCall>
```

Next, the user may acquire a list of all the VMs known to the host: (Note the call takes the session token as the only parameter)

```
>>> all_vms = host.get_resident_VMs(session)['Value']
>>> all_vms
['OpaqueRef:1', 'OpaqueRef:2', 'OpaqueRef:3', 'OpaqueRef:4' ]
```

The VM references here have the form `OpaqueRef:X`, though they may not be that simple in the future, and you should treat them as opaque strings. Once a reference to a VM has been acquired a lifecycle operation may be invoked:

```
>>> xen.VM.start(session, all_vms[3], False)
{'Status': 'Failure', 'ErrorDescription': ['VM_BAD_POWER_STATE', 'Halted', 'Running']}
```

In this case the `start` message has been rejected, because the VM is already running, 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. Finally, here are some examples of using accessors for object fields:

```
>>> xen.VM.get_name_label(session, all_vms[3])['Value']
'SMP'
>>> xen.VM.get_name_description(session, all_vms[3])['Value']
'Debian for Xen'
```

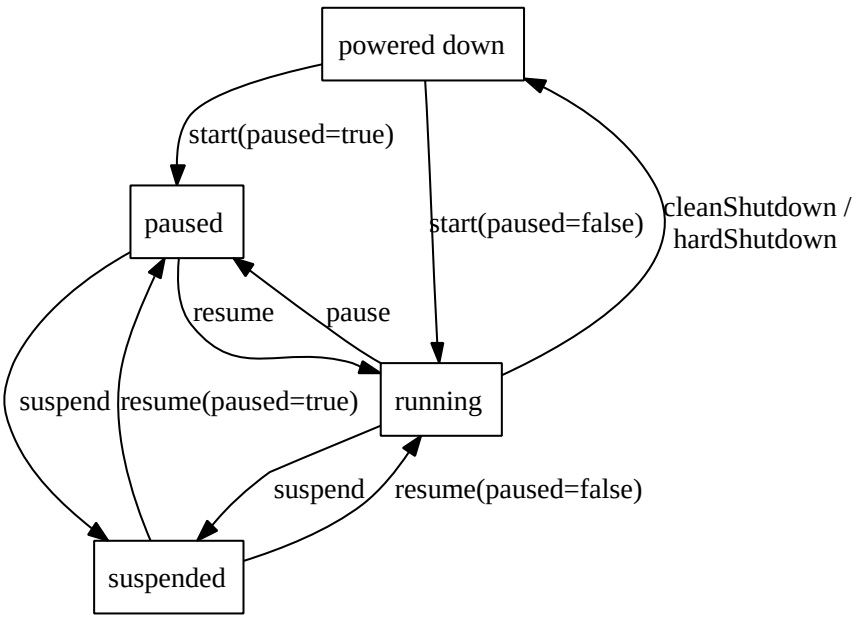


Figure 1.1: VM Lifecycle

1.6 VM Lifecycle

Figure 1.1 shows the states that a VM can be in and the API calls that can be used to move the VM between these states.

1.7 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 the empty string, then paravirtual domain building and booting will be used; otherwise the VM will be loaded as an 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 was 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.

Chapter 2

API Reference

2.1 Classes

The following classes are defined:

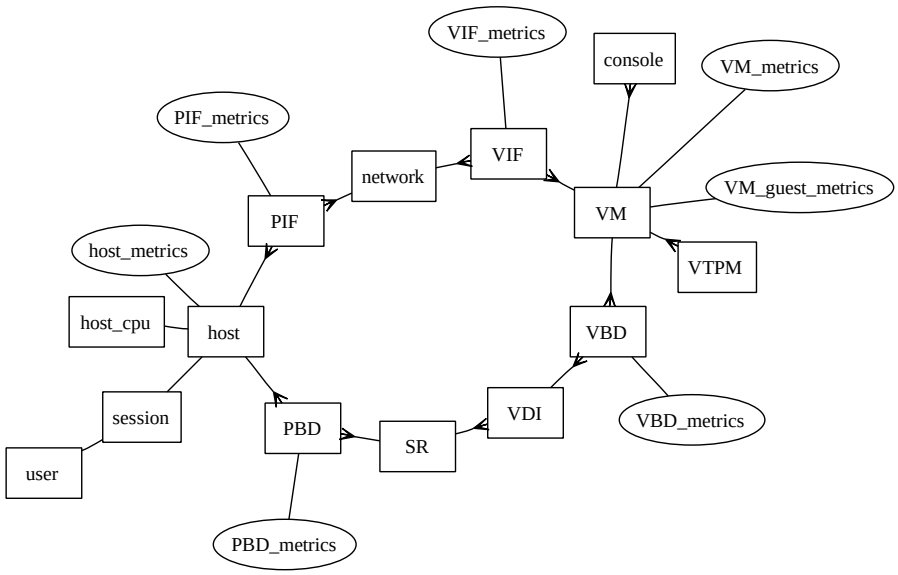
Name	Description
session	A session
task	A long-running asynchronous task
event	Asynchronous event registration and handling
VM	A virtual machine (or 'guest')
VM_metrics	The metrics associated with a VM
VM_guest_metrics	The metrics reported by the guest (as opposed to inferred from outside)
host	A physical host
host_metrics	The metrics associated with a host
host_cpu	A physical CPU
network	A virtual network
VIF	A virtual network interface
VIF_metrics	The metrics associated with a virtual network device
PIF	A physical network interface (note separate VLANs are represented as several PIFs)
PIF_metrics	The metrics associated with a physical network interface
SR	A storage repository
VDI	A virtual disk image
VBD	A virtual block device
VBD_metrics	The metrics associated with a virtual block device
PBD	The physical block devices through which hosts access SRs
crashdump	A VM crashdump
VTPM	A virtual TPM device
console	A console
user	A user of the system
debug	A basic class for testing

2.2 Relationships Between Classes

Fields that are bound together are shown in the following table:

object.field	*object.field*	*relationship*
host.PBDs	PBD.host	many-to-one
SR.PBDs	PBD.SR	many-to-one
VDI.VBDs	VBD.VDI	many-to-one
VDI.crash_dumps	crashdump.VDI	many-to-one
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
PIF.host	host.PIFs	one-to-many
PIF.network	network.PIFs	one-to-many
SR.VDIs	VDI.SR	many-to-one
VTPM.VM	VM.VTPMs	one-to-many
console.VM	VM.consoles	one-to-many
host.resident_VMs	VM.resident_on	many-to-one
host.host_CPUs	host_cpu.host	many-to-one

The following represents bound fields (as specified above) diagramatically, using crows-foot notation to specify one-to-one, one-to-many or many-to-many relationships:



2.2.1 List of bound fields

2.3 Types

2.3.1 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
Ref (object name)	reference to an object of class name

2.3.2 Higher order types

The following type constructors are used:

Type	Description
List (t)	an arbitrary-length list of elements of type t
Map (a → b)	a table mapping values of type a to values of type b

2.3.3 Enumeration types

The following enumeration types are used:

enum event_operation	
add	An object has been created
del	An object has been deleted
mod	An object has been modified

enum console_protocol	
vt100	VT100 terminal
rfb	Remote FrameBuffer protocol (as used in VNC)
rdp	Remote Desktop Protocol

enum vdi_type	
system	a disk that may be replaced on upgrade
user	a disk that is always preserved on upgrade
ephemeral	a disk that may be reformatted on upgrade
suspend	a disk that stores a suspend image
crashdump	a disk that stores VM crashdump information

enum vm_power_state	
Halted	Halted
Paused	Paused
Running	Running
Suspended	Suspended
Unknown	Some other unknown state

enum task_allowed_operations	
Cancel	Cancel

enum task_status_type	
pending	task is in progress
success	task was completed successfully
failure	task has failed
cancelling	task is being cancelled
cancelled	task has been cancelled

enum on_normal_exit	
destroy	destroy the VM state
restart	restart the VM

enum on_crash_behaviour	
destroy	destroy the VM state
coredump_and_destroy	record a coredump and then destroy the VM state
restart	restart the VM
coredump_and_restart	record a coredump and then restart the VM
preserve	leave the crashed VM as-is
rename_restart	rename the crashed VM and start a new copy

enum vbd_mode	
RO	disk is mounted read-only
RW	disk is mounted read-write

enum vbd_type	
CD	VBD will appear to guest as CD
Disk	VBD will appear to guest as disk

2.4 Class: session

2.4.1 Fields for class: session

Name	**session**		
Description	*A session.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`this_host`	host ref	Currently connected host
RO_{run}	`this_user`	user ref	Currently connected user
RO_{run}	`last_active`	int	Timestamp for last time session was active

2.4.2 RPCs associated with class: session

RPC name: login_with_password

Overview: Attempt to authenticate the user, returning a session_id if successful.
Signature:

```
(session ref) login_with_password (string uname, string pwd)
```

Arguments:

type	name	description
`string`	uname	Username for login.
`string`	pwd	Password for login.

Return Type: `session ref`
ID of newly created session

RPC name: logout

Overview: Log out of a session.
Signature:

```
void logout (session_id s)
```

Return Type: `void`

RPC name: get_uuid

Overview: Get the uuid field of the given session.
Signature:

```
string get_uuid (session_id s, session ref self)
```

Arguments:

type	name	description
`session ref`	self	reference to the object

Return Type: `string`

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_id s, session ref self)
```

Arguments:

type	name	description
`session ref`	self	reference to the object

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_id s, session ref self)
```

Arguments:

type	name	description
`session ref`	self	reference to the object

Return Type: `user ref`
value of the field

RPC name: get_last_active

Overview: Get the last_active field of the given session.
Signature:

```
int get_last_active (session_id s, session ref self)
```

Arguments:

type	name	description
`session ref`	self	reference to the object

Return Type: `int`
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_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `session ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given session.
Signature:

```
(session record) get_record (session_id s, session ref self)
```

Arguments:

type	name	description
`session ref`	self	reference to the object

Return Type: `session record`
all fields from the object

2.5 Class: task

2.5.1 Fields for class: task

Name	**task**		
Description	*A long-running asynchronous task.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`name/label`	string	a human-readable name
RO_{run}	`name/description`	string	a notes field containg human-readable description
RO_{run}	`status`	task_status_type	current status of the task
RO_{run}	`session`	session ref	the session that created the task
RO_{run}	`progress`	int	if the task is still pending, this field contains the estimated percentage complete (0-100). If task has completed (successfully or unsuccessfully) this should be 100.
RO_{run}	`type`	string	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.
RO_{run}	`result`	string	if the task has completed successfully, this field contains the result value (either Void or an object reference). Undefined otherwise.
RO_{run}	`error_info`	string Set	if the task has failed, this field contains the set of associated error strings. Undefined otherwise.
RO_{run}	`allowed_operations`	(task_allowed_operations) Set	Operations allowed on this task

2.5.2 RPCs associated with class: task

RPC name: cancel

Overview: Cancel this task. If task.allowed_operations does not contain Cancel, then this will fail with OPERATION_NOT_ALLOWED. The task will show the status 'cancelling', and you should continue to check its status until it shows 'cancelled'. There is no guarantee as to the time within which this task will be cancelled.
Signature:

```
void cancel (session_id s, task ref task)
```

Arguments:

type	name	description
`task ref`	task	The task

Return Type: `void`

Possible Error Codes: `OPERATION_NOT_ALLOWED`

RPC name: get_all

Overview: Return a list of all the tasks known to the system.
Signature:

```
((task ref) Set) get_all (session_id s)
```

Return Type: `(task ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given task.
Signature:

```
string get_uuid (session_id s, task ref self)
```

Arguments:

type	name	description
`task ref`	self	reference to the object

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_id s, task ref self)
```

Arguments:

type	name	description
`task ref`	self	reference to the object

Return Type: `string`
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_id s, task ref self)
```

Arguments:

type	name	description
`task ref`	self	reference to the object

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_id s, task ref self)
```

Arguments:

type	name	description
task ref	self	reference to the object

Return Type: task_status_type
value of the field

RPC name: get_session

Overview: Get the session field of the given task.
Signature:

```
(session ref) get_session (session_id s, task ref self)
```

Arguments:

type	name	description
task ref	self	reference to the object

Return Type: session ref
value of the field

RPC name: get_progress

Overview: Get the progress field of the given task.
Signature:

```
int get_progress (session_id s, task ref self)
```

Arguments:

type	name	description
task ref	self	reference to the object

Return Type: int
value of the field

RPC name: get_type

Overview: Get the type field of the given task.
Signature:

```
string get_type (session_id s, task ref self)
```

Arguments:

type	name	description
task ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_result

Overview: Get the result field of the given task.
Signature:

```
string get_result (session_id s, task ref self)
```

Arguments:

type	name	description
task ref	self	reference to the object

Return Type: string
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_id s, task ref self)
```

Arguments:

type	name	description
task ref	self	reference to the object

Return Type: string Set
value of the field

RPC name: get_allowed_operations

Overview: Get the allowed_operations field of the given task.
Signature:

```
((task_allowed_operations) Set) get_allowed_operations (session_id s, task ref self)
```

Arguments:

type	name	description
`task ref`	self	reference to the object

Return Type: `(task_allowed_operations) Set`
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the task instance with the specified UUID.
Signature:

```
(task ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `task ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given task.
Signature:

```
(task record) get_record (session_id s, task ref self)
```

Arguments:

type	name	description
`task ref`	self	reference to the object

Return Type: `task record`
all fields from the object

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_id s, string label)
```

Arguments:

type	name	description
`string`	label	label of object to return

Return Type: `(task ref) Set`
references to objects with matching names

2.6 Class: event

2.6.1 Fields for class: event

Name	**event**		
Description	*Asynchronous event registration and handling.*		
Quals	Field	Type	Description
RO_{ins}	`id`	int	An ID, monotonically increasing, and local to the current session
RO_{ins}	`timestamp`	datetime	The time at which the event occurred
RO_{ins}	`class`	string	The name of the class of the object that changed
RO_{ins}	`operation`	event_operation	The operation that was performed
RO_{ins}	`ref`	string	A reference to the object that changed
RO_{ins}	`obj_uuid`	string	The uuid of the object that changed

2.6.2 RPCs associated with class: event

RPC name: register

Overview: Registers this session with the event system. Specifying the empty list will register for all classes.
Signature:

```
void register (session_id s, string Set classes)
```

Arguments:

type	name	description
`string Set`	classes	register for events for the indicated classes

Return Type: `void`

RPC name: unregister

Overview: Unregisters this session with the event system.
Signature:

```
void unregister (session_id s, string Set classes)
```

Arguments:

type	name	description
`string Set`	classes	remove this session's registration for the indicated classes

Return Type: `void`

RPC name: next

Overview: Blocking call which returns a (possibly empty) batch of events.
Signature:

```
((event record) Set) next (session_id s)
```

Return Type: `(event record) Set`
the batch of events

Possible Error Codes: `SESSION_NOT_REGISTERED`

2.7 Class: VM

2.7.1 Fields for class: VM

Name	**VM**		
Description	*A virtual machine (or 'guest').*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`power_state`	vm_power_state	Current power state of the machine
RW	`name/label`	string	a human-readable name
RW	`name/description`	string	a notes field containg human-readable description
RW	`user_version`	int	a user version number for this machine
RW	`is_a_template`	bool	true if this is a template. Template VMs can never be started, they are used only for cloning other VMs
RW	`auto_power_on`	bool	true if this VM should be started automatically after host boot
RO_{run}	`suspend_VDI`	VDI ref	The VDI that a suspend image is stored on. (Only has meaning if VM is currently suspended)
RO_{run}	`resident_on`	host ref	the host the VM is currently resident on
RW	`memory/static_max`	int	Statically-set (i.e. absolute) maximum (bytes)
RW	`memory/dynamic_max`	int	Dynamic maximum (bytes)
RW	`memory/dynamic_min`	int	Dynamic minimum (bytes)
RW	`memory/static_min`	int	Statically-set (i.e. absolute) mininum (bytes)
RW	`VCPUs/params`	(string → string) Map	configuration parameters for the selected VCPU policy
RW	`VCPUs/max`	int	Max number of VCPUs
RW	`VCPUs/at_startup`	int	Boot number of VCPUs
RW	`actions/after_shutdown`	on_normal_exit	action to take after the guest has shutdown itself
RW	`actions/after_reboot`	on_normal_exit	action to take after the guest has rebooted itself
RW	`actions/after_crash`	on_crash_behaviour	action to take if the guest crashes
RO_{run}	`consoles`	(console ref) Set	virtual console devices
RO_{run}	`VIFs`	(VIF ref) Set	virtual network interfaces
RO_{run}	`VBDs`	(VBD ref) Set	virtual block devices
RO_{run}	`crash_dumps`	(crashdump ref) Set	crash dumps associated with this VM
RO_{run}	`VTPMs`	(VTPM ref) Set	virtual TPMs
RW	`PV/bootloader`	string	name of or path to bootloader
RW	`PV/kernel`	string	path to the kernel
RW	`PV/ramdisk`	string	path to the initrd
RW	`PV/args`	string	kernel command-line arguments
RW	`PV/bootloader_args`	string	miscellaneous arguments for the bootloader
RW	`HVM/boot_policy`	string	HVM boot policy
RW	`HVM/boot_params`	(string → string) Map	HVM boot params
RW	`platform`	(string → string) Map	platform-specific configuration

RW	PCI_bus	string	PCI bus path for pass-through devices
RW	other_config	(string → string) Map	additional configuration
RO_{run}	domid	int	domain ID (if available, -1 otherwise)
RO_{run}	is_control_domain	bool	true if this is a control domain (domain 0 or a driver domain)
RO_{run}	metrics	VM_metrics ref	metrics associated with this VM
RO_{run}	guest_metrics	VM_guest_metrics ref	metrics associated with the running guest

2.7.2 RPCs associated with class: VM

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_id s, VM ref vm, string new_name)
```

Arguments:

type	name	description
VM ref	vm	The VM to be cloned
string	new_name	The name of the cloned VM

Return Type: VM ref
The ID of the newly created VM.

Possible Error Codes: VM_BAD_POWER_STATE, SR_FULL, OPERATION_NOT_ALLOWED

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_id s, VM ref vm, bool start_paused, bool force)
```

Arguments:

type	name	description
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)

Return Type: void

Possible Error Codes: VM_BAD_POWER_STATE, VM_HVM_REQUIRED, VM_IS_TEMPLATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED

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_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to pause

Return Type: void

Possible Error Codes: VM_BAD_POWER_STATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED, 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_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to unpause

Return Type: void

Possible Error Codes: VM_BAD_POWER_STATE, OPERATION_NOT_ALLOWED, 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).
Once shutdown has been completed perform poweroff action specified in guest configuration.
This can only be called when the specified VM is in the Running state.
Signature:

```
void clean_shutdown (session_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to shutdown

Return Type: void

Possible Error Codes: VM_BAD_POWER_STATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

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).
Once shutdown has been completed perform reboot action specified in guest configuration.
This can only be called when the specified VM is in the Running state.
Signature:

```
void clean_reboot (session_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to shutdown

Return Type: `void`

Possible Error Codes: VM_BAD_POWER_STATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

RPC name: hard_shutdown

Overview: Stop executing the specified VM without attempting a clean shutdown. Then perform poweroff action specified in VM configuration.
Signature:

```
void hard_shutdown (session_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to destroy

Return Type: `void`

Possible Error Codes: VM_BAD_POWER_STATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

RPC name: hard_reboot

Overview: Stop executing the specified VM without attempting a clean shutdown. Then perform reboot action specified in VM configuration.
Signature:

```
void hard_reboot (session_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to reboot

Return Type: `void`

Possible Error Codes: VM_BAD_POWER_STATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

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_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to suspend

Return Type: void

Possible Error Codes: VM_BAD_POWER_STATE, VM_OTHER_OPERATION_IN_PROGRESS, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

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_id s, VM ref vm, bool start_paused, bool force)
```

Arguments:

type	name	description
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)

Return Type: void

Possible Error Codes: VM_BAD_POWER_STATE, OPERATION_NOT_ALLOWED, VM_IS_TEMPLATE

RPC name: set_VCPUs_number_live

Overview: Set this VM's VCPUs/at_startup value, and set the same value on the VM, if running.
Signature:

```
void set_VCPUs_number_live (session_id s, VM ref self, int nvcpu)
```

Arguments:

type	name	description
VM ref	self	The VM
int	nvcpu	The number of VCPUs

Return Type: void

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_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to query

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_id s, VM ref vm)
```

Arguments:

type	name	description
VM ref	vm	The VM to query

Return Type: `string Set`
The allowed values

RPC name: get_all

Overview: Return a list of all the VMs known to the system.
Signature:

```
((VM ref) Set) get_all (session_id s)
```

Return Type: `(VM ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VM.
Signature:

```
string get_uuid (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: `string`

value of the field

RPC name: get_power_state

Overview: Get the power_state field of the given VM.
Signature:

```
(vm_power_state) get_power_state (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: vm_power_state
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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_name_label

Overview: Set the name/label field of the given VM.
Signature:

```
void set_name_label (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_name_description

Overview: Get the name/description field of the given VM.
Signature:

```
string get_name_description (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_name_description

Overview: Set the name/description field of the given VM.
Signature:

```
void set_name_description (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_user_version

Overview: Get the user_version field of the given VM.
Signature:

```
int get_user_version (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: int
value of the field

RPC name: set_user_version

Overview: Set the user_version field of the given VM.
Signature:

```
void set_user_version (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
int	value	New value to set

Return Type: void

RPC name: get_is_a_template

Overview: Get the is_a_template field of the given VM.
Signature:

```
bool get_is_a_template (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: bool
value of the field

RPC name: set_is_a_template

Overview: Set the is_a_template field of the given VM.
Signature:

```
void set_is_a_template (session_id s, VM ref self, bool value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
bool	value	New value to set

Return Type: void

RPC name: get_auto_power_on

Overview: Get the auto_power_on field of the given VM.
Signature:

```
bool get_auto_power_on (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: bool
value of the field

RPC name: set_auto_power_on

Overview: Set the auto_power_on field of the given VM.
Signature:

```
void set_auto_power_on (session_id s, VM ref self, bool value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
bool	value	New value to set

Return Type: `void`

RPC name: get_suspend_VDI

Overview: Get the suspend_VDI field of the given VM.
Signature:

```
(VDI ref) get_suspend_VDI (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: `VDI ref`
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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: `host ref`
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_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: set_memory_static_max

Overview: Set the memory/static_max field of the given VM.
Signature:

```
void set_memory_static_max (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object
`int`	value	New value to set

Return Type: `void`

RPC name: get_memory_dynamic_max

Overview: Get the memory/dynamic_max field of the given VM.
Signature:

```
int get_memory_dynamic_max (session_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: set_memory_dynamic_max

Overview: Set the memory/dynamic_max field of the given VM.
Signature:

```
void set_memory_dynamic_max (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object
`int`	value	New value to set

Return Type: `void`

RPC name: get_memory_dynamic_min

Overview: Get the memory/dynamic_min field of the given VM.
Signature:

```
int get_memory_dynamic_min (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: int
value of the field

RPC name: set_memory_dynamic_min

Overview: Set the memory/dynamic_min field of the given VM.
Signature:

```
void set_memory_dynamic_min (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
int	value	New value to set

Return Type: void

RPC name: get_memory_static_min

Overview: Get the memory/static_min field of the given VM.
Signature:

```
int get_memory_static_min (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: int
value of the field

RPC name: set_memory_static_min

Overview: Set the memory/static_min field of the given VM.
Signature:

```
void set_memory_static_min (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
int	value	New value to set

Return Type: void

RPC name: get_VCPUs_params

Overview: Get the VCPUs/params field of the given VM.
Signature:

```
((string -> string) Map) get_VCPUs_params (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_VCPUs_params

Overview: Set the VCPUs/params field of the given VM.
Signature:

```
void set_VCPUs_params (session_id s, VM ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VM ref self, string key, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, VM ref self, string key)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: get_VCPUs_max

Overview: Get the VCPUs/max field of the given VM.
Signature:

```
int get_VCPUs_max (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: int
value of the field

RPC name: set_VCPUs_max

Overview: Set the VCPUs/max field of the given VM.
Signature:

```
void set_VCPUs_max (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
int	value	New value to set

Return Type: void

RPC name: get_VCPUs_at_startup

Overview: Get the VCPUs/at_startup field of the given VM.
Signature:

```
int get_VCPUs_at_startup (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: int
value of the field

RPC name: set_VCPUs_at_startup

Overview: Set the VCPUs/at_startup field of the given VM.
Signature:

```
void set_VCPUs_at_startup (session_id s, VM ref self, int value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
int	value	New value to set

Return Type: void

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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: on_normal_exit
value of the field

RPC name: set_actions_after_shutdown

Overview: Set the actions/after_shutdown field of the given VM.
Signature:

```
void set_actions_after_shutdown (session_id s, VM ref self, on_normal_exit value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
on_normal_exit	value	New value to set

Return Type: void

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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: on_normal_exit
value of the field

RPC name: set_actions_after_reboot

Overview: Set the actions/after_reboot field of the given VM.
Signature:

```
void set_actions_after_reboot (session_id s, VM ref self, on_normal_exit value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
on_normal_exit	value	New value to set

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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: on_crash_behaviour
value of the field

RPC name: set_actions_after_crash

Overview: Set the actions/after_crash field of the given VM.
Signature:

```
void set_actions_after_crash (session_id s, VM ref self, on_crash_behaviour value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
on_crash_behaviour	value	New value to set

Return Type: void

RPC name: get_consoles

Overview: Get the consoles field of the given VM.
Signature:

```
((console ref) Set) get_consoles (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (console 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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (VIF ref) Set
value of the field

RPC name: get_VBDs

Overview: Get the VBDs field of the given VM.
Signature:

```
((VBD ref) Set) get_VBDs (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (VBD ref) Set
value of the field

RPC name: get_crash_dumps

Overview: Get the crash_dumps field of the given VM.
Signature:

```
((crashdump ref) Set) get_crash_dumps (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (crashdump 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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (VTPM ref) Set
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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_PV_bootloader

Overview: Set the PV/bootloader field of the given VM.
Signature:

```
void set_PV_bootloader (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_PV_kernel

Overview: Get the PV/kernel field of the given VM.
Signature:

```
string get_PV_kernel (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_PV_kernel

Overview: Set the PV/kernel field of the given VM.
Signature:

```
void set_PV_kernel (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_PV_ramdisk

Overview: Get the PV/ramdisk field of the given VM.
Signature:

```
string get_PV_ramdisk (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_PV_ramdisk

Overview: Set the PV/ramdisk field of the given VM.
Signature:

```
void set_PV_ramdisk (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_PV_args

Overview: Get the PV/args field of the given VM.
Signature:

```
string get_PV_args (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_PV_args

Overview: Set the PV/args field of the given VM.
Signature:

```
void set_PV_args (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_PV_bootloader_args

Overview: Get the PV/bootloader_args field of the given VM.
Signature:

```
string get_PV_bootloader_args (session_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_PV_bootloader_args

Overview: Set the PV/bootloader_args field of the given VM.
Signature:

```
void set_PV_bootloader_args (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object
`string`	value	New value to set

Return Type: `void`

RPC name: get_HVM_boot_policy

Overview: Get the HVM/boot_policy field of the given VM.
Signature:

```
string get_HVM_boot_policy (session_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_HVM_boot_policy

Overview: Set the HVM/boot_policy field of the given VM.
Signature:

```
void set_HVM_boot_policy (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object
`string`	value	New value to set

Return Type: `void`

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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_HVM_boot_params

Overview: Set the HVM/boot_params field of the given VM.
Signature:

```
void set_HVM_boot_params (session_id s, VM ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VM ref self, string key, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, VM ref self, string key)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: get_platform

Overview: Get the platform field of the given VM.
Signature:

```
((string -> string) Map) get_platform (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_platform

Overview: Set the platform field of the given VM.
Signature:

```
void set_platform (session_id s, VM ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VM ref self, string key, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, VM ref self, string key)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: get_PCI_bus

Overview: Get the PCI_bus field of the given VM.
Signature:

```
string get_PCI_bus (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_PCI_bus

Overview: Set the PCI_bus field of the given VM.
Signature:

```
void set_PCI_bus (session_id s, VM ref self, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_other_config

Overview: Get the other_config field of the given VM.
Signature:

```
((string -> string) Map) get_other_config (session_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_other_config

Overview: Set the other_config field of the given VM.
Signature:

```
void set_other_config (session_id s, VM ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VM ref self, string key, string value)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, VM ref self, string key)
```

Arguments:

type	name	description
VM ref	self	reference to the object
string	key	Key to remove

Return Type: `void`

RPC name: get_domid

Overview: Get the domid field of the given VM.
Signature:

```
int get_domid (session_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `int`
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_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `bool`
value of the field

RPC name: get_metrics

Overview: Get the metrics field of the given VM.
Signature:

```
(VM_metrics ref) get_metrics (session_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `VM_metrics ref`
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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: VM_guest_metrics ref
value of the field

RPC name: create

Overview: Create a new VM instance, and return its handle.
Signature:

```
(VM ref) create (session_id s, VM record args)
```

Arguments:

type	name	description
VM record	args	All constructor arguments

Return Type: VM ref
reference to the newly created object

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_id s, VM ref self)
```

Arguments:

type	name	description
VM ref	self	reference to the object

Return Type: void

RPC name: get_by_uuid

Overview: Get a reference to the VM instance with the specified UUID.
Signature:

```
(VM ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `VM ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VM.
Signature:

```
(VM record) get_record (session_id s, VM ref self)
```

Arguments:

type	name	description
`VM ref`	self	reference to the object

Return Type: `VM record`
all fields from the object

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_id s, string label)
```

Arguments:

type	name	description
`string`	label	label of object to return

Return Type: `(VM ref) Set`
references to objects with matching names

2.8 Class: VM_metrics

2.8.1 Fields for class: VM_metrics

Name	**VM_metrics**		
Description	*The metrics associated with a VM.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`memory/actual`	int	Guest's actual memory (bytes)
RO_{run}	`VCPUs/number`	int	Current number of VCPUs
RO_{run}	`VCPUs/utilisation`	(int → float) Map	Utilisation for all of guest's current VCPUs
RO_{run}	`last_updated`	datetime	Time at which this information was last updated

2.8.2 RPCs associated with class: VM_metrics

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_id s)
```

Return Type: `(VM_metrics ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VM_metrics.
Signature:

```
string get_uuid (session_id s, VM_metrics ref self)
```

Arguments:

type	name	description
`VM_metrics ref`	self	reference to the object

Return Type: `string`
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_id s, VM_metrics ref self)
```

Arguments:

type	name	description
`VM_metrics ref`	self	reference to the object

Return Type: `int`
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_id s, VM_metrics ref self)
```

Arguments:

type	name	description
`VM_metrics ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: get_VCPUs_utilisation

Overview: Get the VCPUs/utilisation field of the given VM_metrics.
Signature:

```
((int -> float) Map) get_VCPUs_utilisation (session_id s, VM_metrics ref self)
```

Arguments:

type	name	description
`VM_metrics ref`	self	reference to the object

Return Type: `(int → float) Map`
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_id s, VM_metrics ref self)
```

Arguments:

type	name	description
`VM_metrics ref`	self	reference to the object

Return Type: `datetime`
value of the field

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_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: `VM_metrics ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VM_metrics.
Signature:

```
(VM_metrics record) get_record (session_id s, VM_metrics ref self)
```

Arguments:

type	name	description
VM_metrics ref	self	reference to the object

Return Type: `VM_metrics record`
all fields from the object

2.9 Class: VM_guest_metrics

2.9.1 Fields for class: VM_guest_metrics

Name	**VM_guest_metrics**		
Description	*The metrics reported by the guest (as opposed to inferred from outside).*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`os_version`	(string → string) Map	version of the OS
RO_{run}	`PV_drivers_version`	(string → string) Map	version of the PV drivers
RO_{run}	`memory`	(string → string) Map	free/used/total memory
RO_{run}	`disks`	(string → string) Map	disk configuration/free space
RO_{run}	`networks`	(string → string) Map	network configuration
RO_{run}	`other`	(string → string) Map	anything else
RO_{run}	`last_updated`	datetime	Time at which this information was last updated

2.9.2 RPCs associated with class: VM_guest_metrics

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_id s)
```

Return Type: `(VM_guest_metrics ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VM_guest_metrics.
Signature:

```
string get_uuid (session_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
`VM_guest_metrics ref`	self	reference to the object

Return Type: `string`
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_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

Return Type: (string → string) Map
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_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: get_memory

Overview: Get the memory field of the given VM_guest_metrics.
Signature:

```
((string -> string) Map) get_memory (session_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: get_disks

Overview: Get the disks field of the given VM_guest_metrics.
Signature:

```
((string -> string) Map) get_disks (session_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

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_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

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_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

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_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
VM_guest_metrics ref	self	reference to the object

Return Type: datetime
value of the field

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_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `VM_guest_metrics ref`
reference to the object

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_id s, VM_guest_metrics ref self)
```

Arguments:

type	name	description
`VM_guest_metrics ref`	self	reference to the object

Return Type: `VM_guest_metrics record`
all fields from the object

2.10 Class: host

2.10.1 Fields for class: host

Name	**host**		
Description	*A physical host.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RW	`name/label`	string	a human-readable name
RW	`name/description`	string	a notes field containg human-readable description
RO_{run}	`API_version/major`	int	major version number
RO_{run}	`API_version/minor`	int	minor version number
RO_{run}	`API_version/vendor`	string	identification of vendor
RO_{run}	`API_version/vendor_implementation`	(string → string) Map	details of vendor implementation
RO_{run}	`software_version`	(string → string) Map	version strings
RW	`other_config`	(string → string) Map	additional configuration
RO_{run}	`capabilities`	string Set	Xen capabilities
RO_{run}	`supported_bootloaders`	string Set	a list of the bootloaders installed on the machine
RO_{run}	`resident_VMs`	(VM ref) Set	list of VMs currently resident on host
RW	`logging`	(string → string) Map	logging configuration
RO_{run}	`PIFs`	(PIF ref) Set	physical network interfaces
RW	`suspend_image_sr`	SR ref	The SR in which VDIs for suspend images are created
RW	`crash_dump_sr`	SR ref	The SR in which VDIs for crash dumps are created
RO_{run}	`PBDs`	(PBD ref) Set	physical blockdevices
RO_{run}	`host_CPUs`	(host_cpu ref) Set	The physical CPUs on this host
RO_{run}	`metrics`	host_metrics ref	metrics associated with this host

2.10.2 RPCs associated with class: host

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_id s, host ref host)
```

Arguments:

type	name	description
`host ref`	host	The Host to disable

Return Type: `void`

RPC name: enable

Overview: Puts the host into a state in which new VMs can be started.
Signature:

```
void enable (session_id s, host ref host)
```

Arguments:

type	name	description
`host ref`	host	The Host to enable

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_id s, host ref host)
```

Arguments:

type	name	description
`host ref`	host	The Host to shutdown

Return Type: `void`

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_id s, host ref host)
```

Arguments:

type	name	description
`host ref`	host	The Host to reboot

Return Type: `void`

RPC name: dmesg

Overview: Get the host xen dmesg.
Signature:

```
string dmesg (session_id s, host ref host)
```

Arguments:

type	name	description
`host ref`	host	The Host to query

Return Type: `string`
dmesg string

RPC name: list_methods

Overview: List all supported methods.
Signature:

```
(string Set) list_methods (session_id s)
```

Return Type: `string Set`
The name of every supported method.

RPC name: get_all

Overview: Return a list of all the hosts known to the system.
Signature:

```
((host ref) Set) get_all (session_id s)
```

Return Type: `(host ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given host.
Signature:

```
string get_uuid (session_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

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_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_name_label

Overview: Set the name/label field of the given host.
Signature:

```
void set_name_label (session_id s, host ref self, string value)
```

Arguments:

type	name	description
host ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_name_description

Overview: Get the name/description field of the given host.
Signature:

```
string get_name_description (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_name_description

Overview: Set the name/description field of the given host.
Signature:

```
void set_name_description (session_id s, host ref self, string value)
```

Arguments:

type	name	description
host ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_API_version_major

Overview: Get the API_version/major field of the given host.
Signature:

```
int get_API_version_major (session_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

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_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

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_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

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_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

Return Type: `(string → string) Map`
value of the field

RPC name: get_software_version

Overview: Get the software_version field of the given host.
Signature:

```
((string -> string) Map) get_software_version (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (string → string) Map
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_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_other_config

Overview: Set the other_config field of the given host.
Signature:

```
void set_other_config (session_id s, host ref self, (string -> string) Map value)
```

Arguments:

type	name	description
host ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, host ref self, string key, string value)
```

Arguments:

type	name	description
host ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, host ref self, string key)
```

Arguments:

type	name	description
host ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: get_capabilities

Overview: Get the capabilities field of the given host.
Signature:

```
(string Set) get_capabilities (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: string Set
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_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: string Set

value of the field

RPC name: get_resident_VMs

Overview: Get the resident_VMs field of the given host.
Signature:

```
((VM ref) Set) get_resident_VMs (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (VM ref) Set
value of the field

RPC name: get_logging

Overview: Get the logging field of the given host.
Signature:

```
((string -> string) Map) get_logging (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_logging

Overview: Set the logging field of the given host.
Signature:

```
void set_logging (session_id s, host ref self, (string -> string) Map value)
```

Arguments:

type	name	description
host ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, host ref self, string key, string value)
```

Arguments:

type	name	description
host ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, host ref self, string key)
```

Arguments:

type	name	description
host ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: get_PIFs

Overview: Get the PIFs field of the given host.
Signature:

```
((PIF ref) Set) get_PIFs (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (PIF ref) 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_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

Return Type: `SR ref`
value of the field

RPC name: set_suspend_image_sr

Overview: Set the suspend_image_sr field of the given host.
Signature:

```
void set_suspend_image_sr (session_id s, host ref self, SR ref value)
```

Arguments:

type	name	description
`host ref`	self	reference to the object
`SR ref`	value	New value to set

Return Type: `void`

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_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

Return Type: `SR ref`
value of the field

RPC name: set_crash_dump_sr

Overview: Set the crash_dump_sr field of the given host.
Signature:

```
void set_crash_dump_sr (session_id s, host ref self, SR ref value)
```

Arguments:

type	name	description
`host ref`	self	reference to the object
`SR ref`	value	New value to set

Return Type: `void`

RPC name: get_PBDs

Overview: Get the PBDs field of the given host.
Signature:

```
((PBD ref) Set) get_PBDs (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (PBD ref) 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_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: (host_cpu ref) Set
value of the field

RPC name: get_metrics

Overview: Get the metrics field of the given host.
Signature:

```
(host_metrics ref) get_metrics (session_id s, host ref self)
```

Arguments:

type	name	description
host ref	self	reference to the object

Return Type: host_metrics ref
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the host instance with the specified UUID.
Signature:

```
(host ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `host ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given host.
Signature:

```
(host record) get_record (session_id s, host ref self)
```

Arguments:

type	name	description
`host ref`	self	reference to the object

Return Type: `host record`
all fields from the object

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_id s, string label)
```

Arguments:

type	name	description
`string`	label	label of object to return

Return Type: `(host ref) Set`
references to objects with matching names

2.11 Class: host_metrics

2.11.1 Fields for class: host_metrics

Name	**host_metrics**		
Description	*The metrics associated with a host.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`memory/total`	int	Host's total memory (bytes)
RO_{run}	`memory/free`	int	Host's free memory (bytes)
RO_{run}	`last_updated`	datetime	Time at which this information was last updated

2.11.2 RPCs associated with class: host_metrics

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_id s)
```

Return Type: `(host_metrics ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given host_metrics.
Signature:

```
string get_uuid (session_id s, host_metrics ref self)
```

Arguments:

type	name	description
`host_metrics ref`	self	reference to the object

Return Type: `string`
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_id s, host_metrics ref self)
```

Arguments:

type	name	description
`host_metrics ref`	self	reference to the object

Return Type: `int`

value of the field

RPC name: get_memory_free

Overview: Get the memory/free field of the given host_metrics.
Signature:

```
int get_memory_free (session_id s, host_metrics ref self)
```

Arguments:

type	name	description
host_metrics ref	self	reference to the object

Return Type: int
value of the field

RPC name: get_last_updated

Overview: Get the last_updated field of the given host_metrics.
Signature:

```
datetime get_last_updated (session_id s, host_metrics ref self)
```

Arguments:

type	name	description
host_metrics ref	self	reference to the object

Return Type: datetime
value of the field

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_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: host_metrics ref
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given host_metrics.
Signature:

```
(host_metrics record) get_record (session_id s, host_metrics ref self)
```

Arguments:

type	name	description
`host_metrics ref`	self	reference to the object

Return Type: `host_metrics record`
all fields from the object

2.12 Class: host_cpu

2.12.1 Fields for class: host_cpu

Name	**host_cpu**		
Description	*A physical CPU.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`host`	host ref	the host the CPU is in
RO_{run}	`number`	int	the number of the physical CPU within the host
RO_{run}	`vendor`	string	the vendor of the physical CPU
RO_{run}	`speed`	int	the speed of the physical CPU
RO_{run}	`modelname`	string	the model name of the physical CPU
RO_{run}	`stepping`	string	the stepping of the physical CPU
RO_{run}	`flags`	string	the flags of the physical CPU
RO_{run}	`utilisation`	float	the current CPU utilisation

2.12.2 RPCs associated with class: host_cpu

RPC name: get_all

Overview: Return a list of all the host_cpus known to the system.
Signature:

```
((host_cpu ref) Set) get_all (session_id s)
```

Return Type: `(host_cpu ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given host_cpu.
Signature:

```
string get_uuid (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_host

Overview: Get the host field of the given host_cpu.
Signature:

```
(host ref) get_host (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `host ref`
value of the field

RPC name: get_number

Overview: Get the number field of the given host_cpu.
Signature:

```
int get_number (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: get_vendor

Overview: Get the vendor field of the given host_cpu.
Signature:

```
string get_vendor (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_speed

Overview: Get the speed field of the given host_cpu.
Signature:

```
int get_speed (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: get_modelname

Overview: Get the modelname field of the given host_cpu.
Signature:

```
string get_modelname (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
host_cpu ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_stepping

Overview: Get the stepping field of the given host_cpu.
Signature:

```
string get_stepping (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
host_cpu ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_flags

Overview: Get the flags field of the given host_cpu.
Signature:

```
string get_flags (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
host_cpu ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_utilisation

Overview: Get the utilisation field of the given host_cpu.
Signature:

```
float get_utilisation (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `float`
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the host_cpu instance with the specified UUID.
Signature:

```
(host_cpu ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `host_cpu ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given host_cpu.
Signature:

```
(host_cpu record) get_record (session_id s, host_cpu ref self)
```

Arguments:

type	name	description
`host_cpu ref`	self	reference to the object

Return Type: `host_cpu record`
all fields from the object

2.13 Class: network

2.13.1 Fields for class: network

Name	**network**		
Description	*A virtual network.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RW	`name/label`	string	a human-readable name
RW	`name/description`	string	a notes field containg human-readable description
RO_{run}	`VIFs`	(VIF ref) Set	list of connected vifs
RO_{run}	`PIFs`	(PIF ref) Set	list of connected pifs
RW	`other_config`	(string → string) Map	additional configuration

2.13.2 RPCs associated with class: network

RPC name: get_all

Overview: Return a list of all the networks known to the system.
Signature:

```
((network ref) Set) get_all (session_id s)
```

Return Type: `(network ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given network.
Signature:

```
string get_uuid (session_id s, network ref self)
```

Arguments:

type	name	description
`network ref`	self	reference to the object

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_id s, network ref self)
```

Arguments:

type	name	description
`network ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_name_label

Overview: Set the name/label field of the given network.
Signature:

```
void set_name_label (session_id s, network ref self, string value)
```

Arguments:

type	name	description
`network ref`	self	reference to the object
`string`	value	New value to set

Return Type: `void`

RPC name: get_name_description

Overview: Get the name/description field of the given network.
Signature:

```
string get_name_description (session_id s, network ref self)
```

Arguments:

type	name	description
`network ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_name_description

Overview: Set the name/description field of the given network.
Signature:

```
void set_name_description (session_id s, network ref self, string value)
```

Arguments:

type	name	description
`network ref`	self	reference to the object
`string`	value	New value to set

Return Type: `void`

RPC name: get_VIFs

Overview: Get the VIFs field of the given network.
Signature:

```
((VIF ref) Set) get_VIFs (session_id s, network ref self)
```

Arguments:

type	name	description
network ref	self	reference to the object

Return Type: (VIF ref) Set
value of the field

RPC name: get_PIFs

Overview: Get the PIFs field of the given network.
Signature:

```
((PIF ref) Set) get_PIFs (session_id s, network ref self)
```

Arguments:

type	name	description
network ref	self	reference to the object

Return Type: (PIF ref) Set
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_id s, network ref self)
```

Arguments:

type	name	description
network ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_other_config

Overview: Set the other_config field of the given network.
Signature:

```
void set_other_config (session_id s, network ref self, (string -> string) Map value)
```

Arguments:

type	name	description
network ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, network ref self, string key, string value)
```

Arguments:

type	name	description
network ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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 network. If the key is not in that Map, then do nothing.
Signature:

```
void remove_from_other_config (session_id s, network ref self, string key)
```

Arguments:

type	name	description
network ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: create

Overview: Create a new network instance, and return its handle.
Signature:

```
(network ref) create (session_id s, network record args)
```

Arguments:

type	name	description
network record	args	All constructor arguments

Return Type: network ref

reference to the newly created object

RPC name: destroy

Overview: Destroy the specified network instance.
Signature:

```
void destroy (session_id s, network ref self)
```

Arguments:

type	name	description
network ref	self	reference to the object

Return Type: void

RPC name: get_by_uuid

Overview: Get a reference to the network instance with the specified UUID.
Signature:

```
(network ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: network ref
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given network.
Signature:

```
(network record) get_record (session_id s, network ref self)
```

Arguments:

type	name	description
network ref	self	reference to the object

Return Type: network record
all fields from the object

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_id s, string label)
```

Arguments:

type	name	description
`string`	label	label of object to return

Return Type: `(network ref) Set`
references to objects with matching names

2.14 Class: VIF

2.14.1 Fields for class: VIF

Name	**VIF**		
Description	*A virtual network interface.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RW	`device`	string	name of network device as exposed to guest e.g. eth0
RO_{ins}	`network`	network ref	virtual network to which this vif is connected
RO_{ins}	`VM`	VM ref	virtual machine to which this vif is connected
RW	`MAC`	string	ethernet MAC address of virtual interface, as exposed to guest
RW	`MTU`	int	MTU in octets
RO_{run}	`currently_attached`	bool	is the device currently attached (erased on reboot)
RO_{run}	`status_code`	int	error/success code associated with last attach-operation (erased on reboot)
RO_{run}	`status_detail`	string	error/success information associated with last attach-operation status (erased on reboot)
RW	`qos/algorithm_type`	string	QoS algorithm to use
RW	`qos/algorithm_params`	(string → string) Map	parameters for chosen QoS algorithm
RO_{run}	`qos/supported_algorithms`	string Set	supported QoS algorithms for this VIF
RO_{run}	`metrics`	VIF_metrics ref	metrics associated with this VIF

2.14.2 RPCs associated with class: VIF

RPC name: plug

Overview: Hotplug the specified VIF, dynamically attaching it to the running VM.
Signature:

```
void plug (session_id s, VIF ref self)
```

Arguments:

type	name	description
`VIF ref`	self	The VIF to hotplug

Return Type: `void`

RPC name: unplug

Overview: Hot-unplug the specified VIF, dynamically unattaching it from the running VM.
Signature:

```
void unplug (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	The VIF to hot-unplug

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_id s)
```

Return Type: (VIF ref) Set
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VIF.
Signature:

```
string get_uuid (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_device

Overview: Get the device field of the given VIF.
Signature:

```
string get_device (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_device

Overview: Set the device field of the given VIF.
Signature:

```
void set_device (session_id s, VIF ref self, string value)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
string	value	New value to set

Return Type: `void`

RPC name: get_network

Overview: Get the network field of the given VIF.
Signature:

```
(network ref) get_network (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: `network ref`
value of the field

RPC name: get_VM

Overview: Get the VM field of the given VIF.
Signature:

```
(VM ref) get_VM (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: `VM ref`
value of the field

RPC name: get_MAC

Overview: Get the MAC field of the given VIF.
Signature:

```
string get_MAC (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_MAC

Overview: Set the MAC field of the given VIF.
Signature:

```
void set_MAC (session_id s, VIF ref self, string value)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_MTU

Overview: Get the MTU field of the given VIF.
Signature:

```
int get_MTU (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: int
value of the field

RPC name: set_MTU

Overview: Set the MTU field of the given VIF.
Signature:

```
void set_MTU (session_id s, VIF ref self, int value)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
int	value	New value to set

Return Type: void

RPC name: get_currently_attached

Overview: Get the currently_attached field of the given VIF.
Signature:

```
bool get_currently_attached (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: bool
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_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

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_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: string
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_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_qos_algorithm_type

Overview: Set the qos/algorithm_type field of the given VIF.
Signature:

```
void set_qos_algorithm_type (session_id s, VIF ref self, string value)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
string	value	New value to set

Return Type: void

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_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_qos_algorithm_params

Overview: Set the qos/algorithm_params field of the given VIF.
Signature:

```
void set_qos_algorithm_params (session_id s, VIF ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VIF ref self, string key, string value)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, VIF ref self, string key)
```

Arguments:

type	name	description
VIF ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: get_qos_supported_algorithms

Overview: Get the qos/supported_algorithms field of the given VIF.
Signature:

```
(string Set) get_qos_supported_algorithms (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: string Set
value of the field

RPC name: get_metrics

Overview: Get the metrics field of the given VIF.
Signature:

```
(VIF_metrics ref) get_metrics (session_id s, VIF ref self)
```

Arguments:

type	name	description
`VIF ref`	self	reference to the object

Return Type: `VIF_metrics ref`
value of the field

RPC name: create

Overview: Create a new VIF instance, and return its handle.
Signature:

```
(VIF ref) create (session_id s, VIF record args)
```

Arguments:

type	name	description
`VIF record`	args	All constructor arguments

Return Type: `VIF ref`
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified VIF instance.
Signature:

```
void destroy (session_id s, VIF ref self)
```

Arguments:

type	name	description
`VIF ref`	self	reference to the object

Return Type: `void`

RPC name: get_by_uuid

Overview: Get a reference to the VIF instance with the specified UUID.
Signature:

```
(VIF ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `VIF ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VIF.
Signature:

```
(VIF record) get_record (session_id s, VIF ref self)
```

Arguments:

type	name	description
VIF ref	self	reference to the object

Return Type: `VIF record`
all fields from the object

2.15 Class: VIF_metrics

2.15.1 Fields for class: VIF_metrics

Name	**VIF_metrics**		
Description	*The metrics associated with a virtual network device.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`io/read_kbs`	float	Read bandwidth (KiB/s)
RO_{run}	`io/write_kbs`	float	Write bandwidth (KiB/s)
RO_{run}	`last_updated`	datetime	Time at which this information was last updated

2.15.2 RPCs associated with class: VIF_metrics

RPC name: get_all

Overview: Return a list of all the VIF_metrics instances known to the system.
Signature:

```
((VIF_metrics ref) Set) get_all (session_id s)
```

Return Type: `(VIF_metrics ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VIF_metrics.
Signature:

```
string get_uuid (session_id s, VIF_metrics ref self)
```

Arguments:

type	name	description
`VIF_metrics ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_io_read_kbs

Overview: Get the io/read_kbs field of the given VIF_metrics.
Signature:

```
float get_io_read_kbs (session_id s, VIF_metrics ref self)
```

Arguments:

type	name	description
`VIF_metrics ref`	self	reference to the object

Return Type: `float`

value of the field

RPC name: get_io_write_kbs

Overview: Get the io/write_kbs field of the given VIF_metrics.
Signature:

```
float get_io_write_kbs (session_id s, VIF_metrics ref self)
```

Arguments:

type	name	description
VIF_metrics ref	self	reference to the object

Return Type: `float`
value of the field

RPC name: get_last_updated

Overview: Get the last_updated field of the given VIF_metrics.
Signature:

```
datetime get_last_updated (session_id s, VIF_metrics ref self)
```

Arguments:

type	name	description
VIF_metrics ref	self	reference to the object

Return Type: `datetime`
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the VIF_metrics instance with the specified UUID.
Signature:

```
(VIF_metrics ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: `VIF_metrics ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VIF_metrics.
Signature:

```
(VIF_metrics record) get_record (session_id s, VIF_metrics ref self)
```

Arguments:

type	name	description
VIF_metrics ref	self	reference to the object

Return Type: VIF_metrics record
all fields from the object

2.16 Class: PIF

2.16.1 Fields for class: PIF

Name	**PIF**		
Description	*A physical network interface (note separate VLANs are represented as several PIFs).*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RW	`device`	string	machine-readable name of the interface (e.g. eth0)
RO_{ins}	`network`	network ref	virtual network to which this pif is connected
RO_{ins}	`host`	host ref	physical machine to which this pif is connected
RW	`MAC`	string	ethernet MAC address of physical interface
RW	`MTU`	int	MTU in octets
RW	`VLAN`	int	VLAN tag for all traffic passing through this interface
RO_{run}	`metrics`	PIF_metrics ref	metrics associated with this PIF

2.16.2 RPCs associated with class: PIF

RPC name: create_VLAN

Overview: Create a VLAN interface from an existing physical interface.
Signature:

```
(PIF ref) create_VLAN (session_id s, string device, network ref network, host ref host, int VLAN)
```

Arguments:

type	name	description
`string`	device	physical interface on which to crate 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

Return Type: `PIF ref`
The reference of the created PIF object

Possible Error Codes: `VLAN_TAG_INVALID`

RPC name: destroy

Overview: Destroy the interface (provided it is a synthetic interface like a VLAN; fail if it is a physical interface).
Signature:

```
void destroy (session_id s, PIF ref self)
```

Arguments:

type	name	description
`PIF ref`	self	the PIF object to destroy

Return Type: `void`

Possible Error Codes: `PIF_IS_PHYSICAL`

RPC name: get_all

Overview: Return a list of all the PIFs known to the system.
Signature:

```
((PIF ref) Set) get_all (session_id s)
```

Return Type: `(PIF ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given PIF.
Signature:

```
string get_uuid (session_id s, PIF ref self)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_device

Overview: Get the device field of the given PIF.
Signature:

```
string get_device (session_id s, PIF ref self)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_device

Overview: Set the device field of the given PIF.
Signature:

```
void set_device (session_id s, PIF ref self, string value)
```

Arguments:

type	name	description
PIF ref	self	reference to the object
string	value	New value to set

Return Type: `void`

RPC name: get_network

Overview: Get the network field of the given PIF.
Signature:

```
(network ref) get_network (session_id s, PIF ref self)
```

Arguments:

type	name	description
PIF ref	self	reference to the object

Return Type: `network ref`
value of the field

RPC name: get_host

Overview: Get the host field of the given PIF.
Signature:

```
(host ref) get_host (session_id s, PIF ref self)
```

Arguments:

type	name	description
PIF ref	self	reference to the object

Return Type: `host ref`
value of the field

RPC name: get_MAC

Overview: Get the MAC field of the given PIF.
Signature:

```
string get_MAC (session_id s, PIF ref self)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_MAC

Overview: Set the MAC field of the given PIF.
Signature:

```
void set_MAC (session_id s, PIF ref self, string value)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object
`string`	value	New value to set

Return Type: `void`

RPC name: get_MTU

Overview: Get the MTU field of the given PIF.
Signature:

```
int get_MTU (session_id s, PIF ref self)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: set_MTU

Overview: Set the MTU field of the given PIF.
Signature:

```
void set_MTU (session_id s, PIF ref self, int value)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object
`int`	value	New value to set

Return Type: `void`

RPC name: get_VLAN

Overview: Get the VLAN field of the given PIF.
Signature:

```
int get_VLAN (session_id s, PIF ref self)
```

Arguments:

type	name	description
PIF ref	self	reference to the object

Return Type: `int`
value of the field

RPC name: set_VLAN

Overview: Set the VLAN field of the given PIF.
Signature:

```
void set_VLAN (session_id s, PIF ref self, int value)
```

Arguments:

type	name	description
PIF ref	self	reference to the object
int	value	New value to set

Return Type: `void`

RPC name: get_metrics

Overview: Get the metrics field of the given PIF.
Signature:

```
(PIF_metrics ref) get_metrics (session_id s, PIF ref self)
```

Arguments:

type	name	description
PIF ref	self	reference to the object

Return Type: `PIF_metrics 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_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `PIF ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given PIF.
Signature:

```
(PIF record) get_record (session_id s, PIF ref self)
```

Arguments:

type	name	description
`PIF ref`	self	reference to the object

Return Type: `PIF record`
all fields from the object

2.17 Class: PIF_metrics

2.17.1 Fields for class: PIF_metrics

Name	**PIF_metrics**		
Description	*The metrics associated with a physical network interface.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`io/read_kbs`	float	Read bandwidth (KiB/s)
RO_{run}	`io/write_kbs`	float	Write bandwidth (KiB/s)
RO_{run}	`last_updated`	datetime	Time at which this information was last updated

2.17.2 RPCs associated with class: PIF_metrics

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_id s)
```

Return Type: `(PIF_metrics ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given PIF_metrics.
Signature:

```
string get_uuid (session_id s, PIF_metrics ref self)
```

Arguments:

type	name	description
`PIF_metrics ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_io_read_kbs

Overview: Get the io/read_kbs field of the given PIF_metrics.
Signature:

```
float get_io_read_kbs (session_id s, PIF_metrics ref self)
```

Arguments:

type	name	description
`PIF_metrics ref`	self	reference to the object

Return Type: `float`

value of the field

RPC name: get_io_write_kbs

Overview: Get the io/write_kbs field of the given PIF_metrics.
Signature:

```
float get_io_write_kbs (session_id s, PIF_metrics ref self)
```

Arguments:

type	name	description
PIF_metrics ref	self	reference to the object

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_id s, PIF_metrics ref self)
```

Arguments:

type	name	description
PIF_metrics ref	self	reference to the object

Return Type: `datetime`
value of the field

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_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: `PIF_metrics ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given PIF_metrics.
Signature:

```
(PIF_metrics record) get_record (session_id s, PIF_metrics ref self)
```

Arguments:

type	name	description
PIF_metrics ref	self	reference to the object

Return Type: PIF_metrics record
all fields from the object

2.18 Class: SR

2.18.1 Fields for class: SR

Name	**SR**		
Description	*A storage repository.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RW	`name/label`	string	a human-readable name
RW	`name/description`	string	a notes field containg human-readable description
RO_{run}	`VDIs`	(VDI ref) Set	managed virtual disks
RO_{run}	`PBDs`	(PBD ref) Set	physical blockdevices
RO_{run}	`virtual_allocation`	int	sum of virtual_sizes of all VDIs in this storage repository (in bytes)
RO_{run}	`physical_utilisation`	int	physical space currently utilised on this storage repository (in bytes). Note that for sparse disk formats, physical_utilisation may be less than virtual_allocation
RO_{ins}	`physical_size`	int	total physical size of the repository (in bytes)
RO_{ins}	`type`	string	type of the storage repository
RO_{ins}	`content_type`	string	the type of the SR's content, if required (e.g. ISOs)

2.18.2 RPCs associated with class: SR

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_id s)
```

Return Type: `string Set`
the supported SR types

RPC name: get_all

Overview: Return a list of all the SRs known to the system.
Signature:

```
((SR ref) Set) get_all (session_id s)
```

Return Type: `(SR ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given SR.
Signature:

```
string get_uuid (session_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

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_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_name_label

Overview: Set the name/label field of the given SR.
Signature:

```
void set_name_label (session_id s, SR ref self, string value)
```

Arguments:

type	name	description
SR ref	self	reference to the object
string	value	New value to set

Return Type: `void`

RPC name: get_name_description

Overview: Get the name/description field of the given SR.
Signature:

```
string get_name_description (session_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_name_description

Overview: Set the name/description field of the given SR.
Signature:

```
void set_name_description (session_id s, SR ref self, string value)
```

Arguments:

type	name	description
SR ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_VDIs

Overview: Get the VDIs field of the given SR.
Signature:

```
((VDI ref) Set) get_VDIs (session_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: (VDI ref) Set
value of the field

RPC name: get_PBDs

Overview: Get the PBDs field of the given SR.
Signature:

```
((PBD ref) Set) get_PBDs (session_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: (PBD 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_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

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_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: int
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_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: int
value of the field

RPC name: get_type

Overview: Get the type field of the given SR.
Signature:

```
string get_type (session_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: string
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_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the SR instance with the specified UUID.
Signature:

```
(SR ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: SR ref
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given SR.
Signature:

```
(SR record) get_record (session_id s, SR ref self)
```

Arguments:

type	name	description
SR ref	self	reference to the object

Return Type: SR record
all fields from the object

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_id s, string label)
```

Arguments:

type	name	description
`string`	label	label of object to return

Return Type: `(SR ref) Set`
references to objects with matching names

2.19 Class: VDI

2.19.1 Fields for class: VDI

Name	**VDI**		
Description	*A virtual disk image.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RW	`name/label`	string	a human-readable name
RW	`name/description`	string	a notes field containg human-readable description
RO_{ins}	`SR`	SR ref	storage repository in which the VDI resides
RO_{run}	`VBDs`	(VBD ref) Set	list of vbds that refer to this disk
RO_{run}	`crash_dumps`	(crashdump ref) Set	list of crash dumps that refer to this disk
RW	`virtual_size`	int	size of disk as presented to the guest (in bytes). Note that, depending on storage backend type, requested size may not be respected exactly
RO_{run}	`physical_utilisation`	int	amount of physical space that the disk image is currently taking up on the storage repository (in bytes)
RO_{ins}	`type`	vdi_type	type of the VDI
RW	`sharable`	bool	true if this disk may be shared
RW	`read_only`	bool	true if this disk may ONLY be mounted read-only
RW	`other_config`	(string → string) Map	additional configuration

2.19.2 RPCs associated with class: VDI

RPC name: get_all

Overview: Return a list of all the VDIs known to the system.
Signature:

```
((VDI ref) Set) get_all (session_id s)
```

Return Type: `(VDI ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VDI.
Signature:

```
string get_uuid (session_id s, VDI ref self)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object

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_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_name_label

Overview: Set the name/label field of the given VDI.
Signature:

```
void set_name_label (session_id s, VDI ref self, string value)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_name_description

Overview: Get the name/description field of the given VDI.
Signature:

```
string get_name_description (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_name_description

Overview: Set the name/description field of the given VDI.
Signature:

```
void set_name_description (session_id s, VDI ref self, string value)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
string	value	New value to set

Return Type: `void`

RPC name: get_SR

Overview: Get the SR field of the given VDI.
Signature:

```
(SR ref) get_SR (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: `SR ref`
value of the field

RPC name: get_VBDs

Overview: Get the VBDs field of the given VDI.
Signature:

```
((VBD ref) Set) get_VBDs (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: `(VBD ref) Set`
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_id s, VDI ref self)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object

Return Type: `(crashdump 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_id s, VDI ref self)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: set_virtual_size

Overview: Set the virtual_size field of the given VDI.
Signature:

```
void set_virtual_size (session_id s, VDI ref self, int value)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object
`int`	value	New value to set

Return Type: `void`

RPC name: get_physical_utilisation

Overview: Get the physical_utilisation field of the given VDI.
Signature:

```
int get_physical_utilisation (session_id s, VDI ref self)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object

Return Type: `int`
value of the field

RPC name: get_type

Overview: Get the type field of the given VDI.
Signature:

```
(vdi_type) get_type (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: vdi_type
value of the field

RPC name: get_sharable

Overview: Get the sharable field of the given VDI.
Signature:

```
bool get_sharable (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: bool
value of the field

RPC name: set_sharable

Overview: Set the sharable field of the given VDI.
Signature:

```
void set_sharable (session_id s, VDI ref self, bool value)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
bool	value	New value to set

Return Type: void

RPC name: get_read_only

Overview: Get the read_only field of the given VDI.
Signature:

```
bool get_read_only (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: bool
value of the field

RPC name: set_read_only

Overview: Set the read_only field of the given VDI.
Signature:

```
void set_read_only (session_id s, VDI ref self, bool value)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
bool	value	New value to set

Return Type: void

RPC name: get_other_config

Overview: Get the other_config field of the given VDI.
Signature:

```
((string -> string) Map) get_other_config (session_id s, VDI ref self)
```

Arguments:

type	name	description
VDI ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_other_config

Overview: Set the other_config field of the given VDI.
Signature:

```
void set_other_config (session_id s, VDI ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VDI ref self, string key, string value)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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 VDI. If the key is not in that Map, then do nothing.
Signature:

```
void remove_from_other_config (session_id s, VDI ref self, string key)
```

Arguments:

type	name	description
VDI ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: create

Overview: Create a new VDI instance, and return its handle.
Signature:

```
(VDI ref) create (session_id s, VDI record args)
```

Arguments:

type	name	description
VDI record	args	All constructor arguments

Return Type: VDI ref
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified VDI instance.
Signature:

```
void destroy (session_id s, VDI ref self)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object

Return Type: `void`

RPC name: get_by_uuid

Overview: Get a reference to the VDI instance with the specified UUID.
Signature:

```
(VDI ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `VDI ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VDI.
Signature:

```
(VDI record) get_record (session_id s, VDI ref self)
```

Arguments:

type	name	description
`VDI ref`	self	reference to the object

Return Type: `VDI record`
all fields from the object

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_id s, string label)
```

Arguments:

type	name	description
`string`	label	label of object to return

Return Type: `(VDI ref) Set`
references to objects with matching names

2.20 Class: VBD

2.20.1 Fields for class: VBD

Name	**VBD**		
Description	*A virtual block device.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{ins}	`VM`	VM ref	the virtual machine
RO_{ins}	`VDI`	VDI ref	the virtual disk
RO_{run}	`device`	string	device seen by the guest e.g. hda1
RW	`userdevice`	string	user-friendly device name e.g. 0,1,2,etc.
RW	`bootable`	bool	true if this VBD is bootable
RW	`mode`	vbd_mode	the mode the VBD should be mounted with
RW	`type`	vbd_type	how the VBD will appear to the guest (e.g. disk or CD)
RO_{run}	`currently_attached`	bool	is the device currently attached (erased on reboot)
RO_{run}	`status_code`	int	error/success code associated with last attach-operation (erased on reboot)
RO_{run}	`status_detail`	string	error/success information associated with last attach-operation status (erased on reboot)
RW	`qos/algorithm_type`	string	QoS algorithm to use
RW	`qos/algorithm_params`	(string → string) Map	parameters for chosen QoS algorithm
RO_{run}	`qos/supported_algorithms`	string Set	supported QoS algorithms for this VBD
RO_{run}	`metrics`	VBD_metrics ref	metrics associated with this VBD

2.20.2 RPCs associated with class: VBD

RPC name: media_change

Overview: Change the media in the device for CDROM-like devices only. For other devices, detach the VBD and attach a new one.
Signature:

```
void media_change (session_id s, VBD ref vbd, VDI ref vdi)
```

Arguments:

type	name	description
`VBD ref`	vbd	The vbd representing the CDROM-like device
`VDI ref`	vdi	The new VDI to 'insert'

Return Type: `void`

RPC name: plug

Overview: Hotplug the specified VBD, dynamically attaching it to the running VM.
Signature:

```
void plug (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	The VBD to hotplug

Return Type: `void`

RPC name: unplug

Overview: Hot-unplug the specified VBD, dynamically unattaching it from the running VM.
Signature:

```
void unplug (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	The VBD to hot-unplug

Return Type: `void`

RPC name: get_all

Overview: Return a list of all the VBDs known to the system.
Signature:

```
((VBD ref) Set) get_all (session_id s)
```

Return Type: `(VBD ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VBD.
Signature:

```
string get_uuid (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_VM

Overview: Get the VM field of the given VBD.
Signature:

```
(VM ref) get_VM (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: VM ref
value of the field

RPC name: get_VDI

Overview: Get the VDI field of the given VBD.
Signature:

```
(VDI ref) get_VDI (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: VDI ref
value of the field

RPC name: get_device

Overview: Get the device field of the given VBD.
Signature:

```
string get_device (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: string
value of the field

RPC name: get_userdevice

Overview: Get the userdevice field of the given VBD.
Signature:

```
string get_userdevice (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_userdevice

Overview: Set the userdevice field of the given VBD.
Signature:

```
void set_userdevice (session_id s, VBD ref self, string value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
string	value	New value to set

Return Type: void

RPC name: get_bootable

Overview: Get the bootable field of the given VBD.
Signature:

```
bool get_bootable (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: bool
value of the field

RPC name: set_bootable

Overview: Set the bootable field of the given VBD.
Signature:

```
void set_bootable (session_id s, VBD ref self, bool value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
bool	value	New value to set

Return Type: void

RPC name: get_mode

Overview: Get the mode field of the given VBD.
Signature:

```
(vbd_mode) get_mode (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: vbd_mode
value of the field

RPC name: set_mode

Overview: Set the mode field of the given VBD.
Signature:

```
void set_mode (session_id s, VBD ref self, vbd_mode value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
vbd_mode	value	New value to set

Return Type: void

RPC name: get_type

Overview: Get the type field of the given VBD.
Signature:

```
(vbd_type) get_type (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: vbd_type
value of the field

RPC name: set_type

Overview: Set the type field of the given VBD.
Signature:

```
void set_type (session_id s, VBD ref self, vbd_type value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
vbd_type	value	New value to set

Return Type: void

RPC name: get_currently_attached

Overview: Get the currently_attached field of the given VBD.
Signature:

```
bool get_currently_attached (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: bool
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_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

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_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: string
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_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: string
value of the field

RPC name: set_qos_algorithm_type

Overview: Set the qos/algorithm_type field of the given VBD.
Signature:

```
void set_qos_algorithm_type (session_id s, VBD ref self, string value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
string	value	New value to set

Return Type: void

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_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_qos_algorithm_params

Overview: Set the qos/algorithm_params field of the given VBD.
Signature:

```
void set_qos_algorithm_params (session_id s, VBD ref self, (string -> string) Map value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
(string → string) Map	value	New value to set

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_id s, VBD ref self, string key, string value)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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_id s, VBD ref self, string key)
```

Arguments:

type	name	description
VBD ref	self	reference to the object
string	key	Key to remove

Return Type: void

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_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: string Set

value of the field

RPC name: get_metrics

Overview: Get the metrics field of the given VBD.
Signature:

```
(VBD_metrics ref) get_metrics (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: VBD_metrics ref
value of the field

RPC name: create

Overview: Create a new VBD instance, and return its handle.
Signature:

```
(VBD ref) create (session_id s, VBD record args)
```

Arguments:

type	name	description
VBD record	args	All constructor arguments

Return Type: VBD ref
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified VBD instance.
Signature:

```
void destroy (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: void

RPC name: get_by_uuid

Overview: Get a reference to the VBD instance with the specified UUID.
Signature:

```
(VBD ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: `VBD ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VBD.
Signature:

```
(VBD record) get_record (session_id s, VBD ref self)
```

Arguments:

type	name	description
VBD ref	self	reference to the object

Return Type: `VBD record`
all fields from the object

2.21 Class: VBD_metrics

2.21.1 Fields for class: VBD_metrics

Name	**VBD_metrics**		
Description	*The metrics associated with a virtual block device.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`io/read_kbs`	float	Read bandwidth (KiB/s)
RO_{run}	`io/write_kbs`	float	Write bandwidth (KiB/s)
RO_{run}	`last_updated`	datetime	Time at which this information was last updated

2.21.2 RPCs associated with class: VBD_metrics

RPC name: get_all

Overview: Return a list of all the VBD_metrics instances known to the system.
Signature:

```
((VBD_metrics ref) Set) get_all (session_id s)
```

Return Type: `(VBD_metrics ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given VBD_metrics.
Signature:

```
string get_uuid (session_id s, VBD_metrics ref self)
```

Arguments:

type	name	description
`VBD_metrics ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_io_read_kbs

Overview: Get the io/read_kbs field of the given VBD_metrics.
Signature:

```
float get_io_read_kbs (session_id s, VBD_metrics ref self)
```

Arguments:

type	name	description
`VBD_metrics ref`	self	reference to the object

Return Type: `float`

value of the field

RPC name: get_io_write_kbs

Overview: Get the io/write_kbs field of the given VBD_metrics.
Signature:

```
float get_io_write_kbs (session_id s, VBD_metrics ref self)
```

Arguments:

type	name	description
VBD_metrics ref	self	reference to the object

Return Type: `float`
value of the field

RPC name: get_last_updated

Overview: Get the last_updated field of the given VBD_metrics.
Signature:

```
datetime get_last_updated (session_id s, VBD_metrics ref self)
```

Arguments:

type	name	description
VBD_metrics ref	self	reference to the object

Return Type: `datetime`
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the VBD_metrics instance with the specified UUID.
Signature:

```
(VBD_metrics ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: `VBD_metrics ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given VBD_metrics.
Signature:

```
(VBD_metrics record) get_record (session_id s, VBD_metrics ref self)
```

Arguments:

type	name	description
VBD_metrics ref	self	reference to the object

Return Type: VBD_metrics record
all fields from the object

2.22 Class: PBD

2.22.1 Fields for class: PBD

Name	**PBD**		
Description	*The physical block devices through which hosts access SRs.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{ins}	`host`	host ref	physical machine on which the pbd is available
RO_{ins}	`SR`	SR ref	the storage repository that the pbd realises
RO_{ins}	`device_config`	(string → string) Map	a config string to string map that is provided to the host's SR-backend-driver
RO_{run}	`currently_attached`	bool	is the SR currently attached on this host?

2.22.2 RPCs associated with class: PBD

RPC name: get_all

Overview: Return a list of all the PBDs known to the system.
Signature:

```
((PBD ref) Set) get_all (session_id s)
```

Return Type: `(PBD ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given PBD.
Signature:

```
string get_uuid (session_id s, PBD ref self)
```

Arguments:

type	name	description
`PBD ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_host

Overview: Get the host field of the given PBD.
Signature:

```
(host ref) get_host (session_id s, PBD ref self)
```

Arguments:

type	name	description
PBD ref	self	reference to the object

Return Type: host ref
value of the field

RPC name: get_SR

Overview: Get the SR field of the given PBD.
Signature:

```
(SR ref) get_SR (session_id s, PBD ref self)
```

Arguments:

type	name	description
PBD ref	self	reference to the object

Return Type: SR ref
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_id s, PBD ref self)
```

Arguments:

type	name	description
PBD ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: get_currently_attached

Overview: Get the currently_attached field of the given PBD.
Signature:

```
bool get_currently_attached (session_id s, PBD ref self)
```

Arguments:

type	name	description
PBD ref	self	reference to the object

Return Type: bool
value of the field

RPC name: create

Overview: Create a new PBD instance, and return its handle.
Signature:

```
(PBD ref) create (session_id s, PBD record args)
```

Arguments:

type	name	description
PBD record	args	All constructor arguments

Return Type: PBD ref
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified PBD instance.
Signature:

```
void destroy (session_id s, PBD ref self)
```

Arguments:

type	name	description
PBD ref	self	reference to the object

Return Type: void

RPC name: get_by_uuid

Overview: Get a reference to the PBD instance with the specified UUID.
Signature:

```
(PBD ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
string	uuid	UUID of object to return

Return Type: PBD ref
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given PBD.
Signature:

```
(PBD record) get_record (session_id s, PBD ref self)
```

Arguments:

type	name	description
`PBD ref`	self	reference to the object

Return Type: `PBD record`
all fields from the object

2.23 Class: crashdump

2.23.1 Fields for class: crashdump

Name	**crashdump**		
Description	*A VM crashdump.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{ins}	`VM`	VM ref	the virtual machine
RO_{ins}	`VDI`	VDI ref	the virtual disk

2.23.2 RPCs associated with class: crashdump

RPC name: destroy

Overview: Destroy the specified crashdump.
Signature:

```
void destroy (session_id s, crashdump ref self)
```

Arguments:

type	name	description
`crashdump ref`	self	The crashdump to destroy

Return Type: `void`

RPC name: get_all

Overview: Return a list of all the crashdumps known to the system.
Signature:

```
((crashdump ref) Set) get_all (session_id s)
```

Return Type: `(crashdump ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given crashdump.
Signature:

```
string get_uuid (session_id s, crashdump ref self)
```

Arguments:

type	name	description
`crashdump ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_VM

Overview: Get the VM field of the given crashdump.
Signature:

```
(VM ref) get_VM (session_id s, crashdump ref self)
```

Arguments:

type	name	description
`crashdump ref`	self	reference to the object

Return Type: `VM ref`
value of the field

RPC name: get_VDI

Overview: Get the VDI field of the given crashdump.
Signature:

```
(VDI ref) get_VDI (session_id s, crashdump ref self)
```

Arguments:

type	name	description
`crashdump ref`	self	reference to the object

Return Type: `VDI ref`
value of the field

RPC name: get_by_uuid

Overview: Get a reference to the crashdump instance with the specified UUID.
Signature:

```
(crashdump ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `crashdump ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given crashdump.
Signature:

```
(crashdump record) get_record (session_id s, crashdump ref self)
```

Arguments:

type	name	description
`crashdump ref`	self	reference to the object

Return Type: `crashdump record`
all fields from the object

2.24 Class: VTPM

2.24.1 Fields for class: VTPM

Name	**VTPM**		
Description	*A virtual TPM device.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{ins}	`VM`	VM ref	the virtual machine
RO_{ins}	`backend`	VM ref	the domain where the backend is located

2.24.2 RPCs associated with class: VTPM

RPC name: get_uuid

Overview: Get the uuid field of the given VTPM.
Signature:

```
string get_uuid (session_id s, VTPM ref self)
```

Arguments:

type	name	description
`VTPM ref`	self	reference to the object

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_id s, VTPM ref self)
```

Arguments:

type	name	description
`VTPM ref`	self	reference to the object

Return Type: `VM ref`
value of the field

RPC name: get_backend

Overview: Get the backend field of the given VTPM.
Signature:

```
(VM ref) get_backend (session_id s, VTPM ref self)
```

Arguments:

type	name	description
`VTPM ref`	self	reference to the object

Return Type: `VM ref`
value of the field

RPC name: create

Overview: Create a new VTPM instance, and return its handle.
Signature:

```
(VTPM ref) create (session_id s, VTPM record args)
```

Arguments:

type	name	description
`VTPM record`	args	All constructor arguments

Return Type: `VTPM ref`
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified VTPM instance.
Signature:

```
void destroy (session_id s, VTPM ref self)
```

Arguments:

type	name	description
`VTPM ref`	self	reference to the object

Return Type: `void`

RPC name: get_by_uuid

Overview: Get a reference to the VTPM instance with the specified UUID.
Signature:

```
(VTPM ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

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_id s, VTPM ref self)
```

Arguments:

type	name	description
`VTPM ref`	self	reference to the object

Return Type: `VTPM record`
all fields from the object

2.25 Class: console

2.25.1 Fields for class: console

Name	**console**		
Description	*A console.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{run}	`protocol`	console_protocol	the protocol used by this console
RO_{run}	`location`	string	URI for the console service
RO_{run}	`VM`	VM ref	VM to which this console is attached
RW	`other_config`	(string → string) Map	additional configuration

2.25.2 RPCs associated with class: console

RPC name: get_all

Overview: Return a list of all the consoles known to the system.
Signature:

```
((console ref) Set) get_all (session_id s)
```

Return Type: `(console ref) Set`
references to all objects

RPC name: get_uuid

Overview: Get the uuid field of the given console.
Signature:

```
string get_uuid (session_id s, console ref self)
```

Arguments:

type	name	description
`console ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_protocol

Overview: Get the protocol field of the given console.
Signature:

```
(console_protocol) get_protocol (session_id s, console ref self)
```

Arguments:

type	name	description
`console ref`	self	reference to the object

Return Type: `console_protocol`

value of the field

RPC name: get_location

Overview: Get the location field of the given console.
Signature:

```
string get_location (session_id s, console ref self)
```

Arguments:

type	name	description
console ref	self	reference to the object

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_id s, console ref self)
```

Arguments:

type	name	description
console ref	self	reference to the object

Return Type: VM ref
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_id s, console ref self)
```

Arguments:

type	name	description
console ref	self	reference to the object

Return Type: (string → string) Map
value of the field

RPC name: set_other_config

Overview: Set the other_config field of the given console.
Signature:

```
void set_other_config (session_id s, console ref self, (string -> string) Map value)
```

Arguments:

type	name	description
console ref	self	reference to the object
(string → string) Map	value	New value to set

Return Type: void

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_id s, console ref self, string key, string value)
```

Arguments:

type	name	description
console ref	self	reference to the object
string	key	Key to add
string	value	Value to add

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 console. If the key is not in that Map, then do nothing.
Signature:

```
void remove_from_other_config (session_id s, console ref self, string key)
```

Arguments:

type	name	description
console ref	self	reference to the object
string	key	Key to remove

Return Type: void

RPC name: create

Overview: Create a new console instance, and return its handle.
Signature:

```
(console ref) create (session_id s, console record args)
```

Arguments:

type	name	description
`console record`	args	All constructor arguments

Return Type: `console ref`
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified console instance.
Signature:

```
void destroy (session_id s, console ref self)
```

Arguments:

type	name	description
`console ref`	self	reference to the object

Return Type: `void`

RPC name: get_by_uuid

Overview: Get a reference to the console instance with the specified UUID.
Signature:

```
(console ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `console ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given console.
Signature:

```
(console record) get_record (session_id s, console ref self)
```

Arguments:

type	name	description
`console ref`	self	reference to the object

Return Type: `console record`
all fields from the object

2.26 Class: user

2.26.1 Fields for class: user

Name	**user**		
Description	*A user of the system.*		
Quals	Field	Type	Description
RO_{run}	`uuid`	string	unique identifier/object reference
RO_{ins}	`short_name`	string	short name (e.g. userid)
RW	`fullname`	string	full name

2.26.2 RPCs associated with class: user

RPC name: get_uuid

Overview: Get the uuid field of the given user.
Signature:

```
string get_uuid (session_id s, user ref self)
```

Arguments:

type	name	description
`user ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_short_name

Overview: Get the short_name field of the given user.
Signature:

```
string get_short_name (session_id s, user ref self)
```

Arguments:

type	name	description
`user ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: get_fullname

Overview: Get the fullname field of the given user.
Signature:

```
string get_fullname (session_id s, user ref self)
```

Arguments:

type	name	description
`user ref`	self	reference to the object

Return Type: `string`
value of the field

RPC name: set_fullname

Overview: Set the fullname field of the given user.
Signature:

```
void set_fullname (session_id s, user ref self, string value)
```

Arguments:

type	name	description
`user ref`	self	reference to the object
`string`	value	New value to set

Return Type: `void`

RPC name: create

Overview: Create a new user instance, and return its handle.
Signature:

```
(user ref) create (session_id s, user record args)
```

Arguments:

type	name	description
`user record`	args	All constructor arguments

Return Type: `user ref`
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified user instance.
Signature:

```
void destroy (session_id s, user ref self)
```

Arguments:

type	name	description
`user ref`	self	reference to the object

Return Type: `void`

RPC name: get_by_uuid

Overview: Get a reference to the user instance with the specified UUID.
Signature:

```
(user ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `user ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given user.
Signature:

```
(user record) get_record (session_id s, user ref self)
```

Arguments:

type	name	description
`user ref`	self	reference to the object

Return Type: `user record`
all fields from the object

2.27 Class: debug

2.27.1 Fields for class: debug

Class debug has no fields.

2.27.2 RPCs associated with class: debug

RPC name: return_failure

Overview: Return an API 'successful' failure.
Signature:

```
void return_failure (session_id s)
```

Return Type: `void`

RPC name: get_all

Overview: Return a list of all the debugs known to the system.
Signature:

```
((debug ref) Set) get_all (session_id s)
```

Return Type: `(debug ref) Set`
references to all objects

RPC name: create

Overview: Create a new debug instance, and return its handle.
Signature:

```
(debug ref) create (session_id s, debug record args)
```

Arguments:

type	name	description
`debug record`	args	All constructor arguments

Return Type: `debug ref`
reference to the newly created object

RPC name: destroy

Overview: Destroy the specified debug instance.
Signature:

```
void destroy (session_id s, debug ref self)
```

Arguments:

type	name	description
`debug ref`	self	reference to the object

Return Type: `void`

RPC name: get_by_uuid

Overview: Get a reference to the debug instance with the specified UUID.
Signature:

```
(debug ref) get_by_uuid (session_id s, string uuid)
```

Arguments:

type	name	description
`string`	uuid	UUID of object to return

Return Type: `debug ref`
reference to the object

RPC name: get_record

Overview: Get a record containing the current state of the given debug.
Signature:

```
(debug record) get_record (session_id s, debug ref self)
```

Arguments:

type	name	description
`debug ref`	self	reference to the object

Return Type: `debug record`
all fields from the object

2.28 Error Handling

When a low-level transport error occurs, or a request is malformed at the HTTP or XML-RPC level, the server may send an XML-RPC Fault 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:

```
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
            <name>faultCode</name>
            <value><int>-1</int></value>
          </member>
          <member>
            <name>faultString</name>
            <value><string>Malformed request</string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

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:

```
<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>eSpeil Inc.</value>
          <value>eSpeil Incorporated</value>
        </data>
      </array>
    </value>
  </member>
</struct>
```

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.
Each possible error code is documented in the following section.

2.28.1 Error Codes

DEVICE_ALREADY_ATTACHED

The device is already attached to a VM

Signature:

```
DEVICE_ALREADY_ATTACHED(vbd)
```

DEVICE_ALREADY_DETACHED

The device is not currently attached

Signature:

```
DEVICE_ALREADY_DETACHED(vbd)
```

DEVICE_ALREADY_EXISTS

A device with the name given already exists on the selected VM

Signature:

```
DEVICE_ALREADY_EXISTS(device)
```

EVENTS_LOST

Some events have been lost from the queue and cannot be retrieved.

No parameters.

HANDLE_INVALID

You gave an invalid handle. 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)
```

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)
```

INVALID_DEVICE

The device name is invalid

Signature:

```
INVALID_DEVICE(device)
```

INVALID_PATCH

The uploaded patch file is invalid

No parameters.

INVALID_VALUE

The value given is invalid

Signature:

```
INVALID_VALUE(field, value)
```

MAP_DUPLICATE_KEY

You tried to add a key-value pair to a map, but that key is already there. The key, current value, and the new value that you tried to set are all echoed.

Signature:

```
MAP_DUPLICATE_KEY(key, current value, new value)
```

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)
```

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)
```

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)
```

NETWORK_CONTAINS_PIF

The network contains active PIFs and cannot be deleted

Signature:

```
NETWORK_CONTAINS_PIF(pifs)
```

NETWORK_CONTAINS_VIF

The network contains active VIFs and cannot be deleted

Signature:

```
NETWORK_CONTAINS_VIF(vifs)
```

NOT_IMPLEMENTED

The function is not implemented

Signature:

```
NOT_IMPLEMENTED(function)
```

OPERATION_NOT_ALLOWED

You attempted an operation that was not allowed.

No parameters.

PATCH_ALREADY_APPLIED

This patch has already been applied

Signature:

```
PATCH_ALREADY_APPLIED(patch)
```

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)
```

PIF_VLAN_EXISTS

You tried to create a PIF, but it already exists

Signature:

```
PIF_VLAN_EXISTS(PIF)
```

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 handle. 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)
```

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)
```

SR_FULL

The SR is full. Requested new size exceeds the maximum size

Signature:

```
SR_FULL(requested, maximum)
```

SR_HAS_PBD

The SR is still connected to a host via a PBD. It cannot be destroyed.

Signature:

```
SR_HAS_PBD(sr)
```

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)
```

VLAN_TAG_INVALID

You tried to create a VLAN, but the tag you gave was invalid – it mmust be between 0 and 4095. The parameter echoes the VLAN tag you gave.

Signature:

```
VLAN_TAG_INVALID(VLAN)
```

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)
```

VM_HVM_REQUIRED

HVM is required for this operation

Signature:

```
VM_HVM_REQUIRED(vm)
```

VM_IS_TEMPLATE

The operation attempted is not valid for a template VM

Signature:

```
VM_IS_TEMPLATE(vm)
```

VM_OTHER_OPERATION_IN_PROGRESS

Another operation involving the VM is currently in progress

Signature:

```
VM_OTHER_OPERATION_IN_PROGRESS(vm)
```

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)
```

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)
```

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 incompatable 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)
```

Chapter 3

GNU Free Documentation License

Version 1.2, November 2002
Copyright ©2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts,

you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.
You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.
You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.